UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)
UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech

Filtratge de dades per a Support Vector Machines

Treball de fi de grau

Grau en Enginyeria Informàtica
Computació

Autor: David Morán Pomés
Tutor: Lluís A. Belanche Muñoz
Computer Science
22/01/2018

Índex

Llista de taules

Llista de figures

Resum

Les *Support Vector Machines* (SVM) són conjunt de tècniques de *Machine Learning* que poden competir perfectament amb les Xarxes Neuronals Artificials (ANN) a nivell de resultats, però que tenen l'inconvenient de tenir un temps d'entrenament molt més elevat que aquestes. El seu particular mecanisme d'entrenament permet que, utilitzant un subconjunt adequat de les dades per a l'entrenament, s'assoleixin pràcticament els mateixos resultats que si s'utilitzés la totalitat de les dades disponibles. La tria d'aquest subconjunt adequat, però, no és en absolut un problema trivial.

És per aquesta raó que hem desenvolupat un nou algoritme, anomenat **Filtratge per Puresa**. Aquest algoritme ha estat dissenyat per a filtrar les dades disponibles per tal d'oferir a la SVM aquelles que siguin més rellevants per al seu entrenament. L'algoritme s'ha dissenyat específicament per a treballar amb SVMs que realitzen tasques de classificació binària, que són aquelles en que s'ha d'identificar a quina classe pertany un nou objecte d'entre les dues disponibles.

L'algoritme proposat té un conjunt de paràmetres que ens permeten regular-ne la complexitat espacial i temporal, adaptant-lo a les necessitats i possibilitats de cada entorn on s'hagi d'executar. Un paràmetre especificat per l'usuari, la **puresa** de les dades filtrades, permet regular la mida del conjunt de dades filtrades de manera indirecta, tot i que l'algoritme s'ha adaptat també per a que l'usuari pugui indicar-ne el nombre de dades de manera directa.

A més, diverses de les fases de l'algoritme són configurables, en el sentit que podem utilitzar diferents mètodes per a realitzar una mateixa tasca. Dins d'aquest treball se n'exposaran diversos exemples, però aquests es poden ampliar amb qualsevol algoritme que realitzi la tasca corresponent.

Amb l'aplicació d'aquest algoritme a datasets reals, s'han aconseguit reduccions de dades de fins al 80%, sense que es produeixi una gran pèrdua de qualitat en la classificació. Aquests resultats són molt prometedors, i suggereixen que pot ser interessant estudiar el comportament d'aquest algoritme en un altres datasets, i valorar si és possible adaptar-lo a la resolució d'altres problemes de Machine Learning, tals com la classificació múltiple o la regressió.

Resumen

Las *Support Vector Machines* (SVM) son un conjunto de técnicas de *Machine Learning* que pueden competir perfectamente con las Redes Neuronales Artificiales (ANN) a nivel de resultados, pero que tienen el inconveniente de tener un tiempo de entrenamiento mucho más elevado que éstas. Su particular mecanismo de entrenamiento permite que, utilizando un subconjunto adecuado de datos para el entrenamiento, se consigan prácticamente los mismos resultados que si se utilizara la totalidad de los datos disponibles. La elección de éste subconjunto, pero, no es en absoluto un problema trivial.

Es por ésta razón que hemos desarrollado un nuevo algoritmo, llamado **Filtrado por pureza**. Éste algoritmo ha sido diseñado para filtrar los datos disponibles por tal de ofrecer a la SVM aquellos datos que sean más relevantes para su entrenamiento. El algoritmo ha sido diseñado específicamente para trabajar con SVMs que realizan tareas de clasificación binaria, que son aquellas en las que hay que identificar a que clase pertenece un nuevo objeto de entre las dos disponibles.

El algoritmo propuesto tiene un conjunto de parámetros que nos permiten regular su complejidad espacial y temporal, adaptándolo a las necesidades y posibilidades de cada entorno donde se tenga que ejecutar. Un parámetro especificado por el usuario, la **pureza** de los datos filtrados, permite regular el tamaño del conjunto de datos filtrados de forma indirecta, aunque el algoritmo se ha adaptado también para que el usuario pueda indicar el número de datos de forma directa.

Además, varias de las fases del algoritmo son configurables, en el sentido que podemos utilizar diferentes métodos para realizar una misma tarea. Dentro de éste trabajo, se expondrán varios ejemplos, pero éstos se pueden ampliar con cualquier algoritmo que realice el trabajo correspondiente.

Con la aplicación de éste algoritmo a datasets reales, se han conseguido reducciones de datos de hasta el 80%, sin que se produzca una gran pérdida de calidad en la clasificación. Éstos resultados son muy prometedores, y sugieren que puede ser interesante estudiar el comportamiento de éste algoritmo en otros datasets, y valorar si es posible adaptarlo a la resolución de otros problemas de Machine Learning, tales como la clasificación múltiple o la regresión.

Abstract

Support Vector Machines (SVM) are a set of Machine Learning methods that achieve levels of accuracy comparable to those achieved by Artificial Neural Networks (ANN), but they are also slower to train. SVM's particular training method offers the possibility to train an SVM using only a specific subset of the data and achieving almost the same results as if all data were used. The problem of choosing this specific subset of data, however, is not trivial at all.

This is the reason why we developed a new algorithm, called **Purity Filtering**. This algorithm has been designed to filter the training data of the SVM in order to choose the subset of the data that is more relevant to the training phase. The algorithm has been specifically designed to work with binary classification SVMs, whose task in to identify to which class does a new object belong.

The proposed algorithm has a set of parameters that allows us to regulate spatial and temporal complexity, adapting its execution to the needs and possibilities of each environment. A user-specified parameter, the **purity** of the filtered data, is used to indirectly regulate the number of filtered data, even though the algorithm has also been adapted to let the user directly specify the number of filtered data.

Furthermore, some of the algorithm phases can be set up by the user, in the sense that some different methods can be used to do the same task. In this project some examples will be given, but those can be expanded by any other algorithm which also performs the same task correctly.

By using this algorithm with real datasets, reductions up to 80% of the data were achieved with no major loss on the quality of the classification. Results are very promising, and suggest that further study of the behavior of this algorithm when used with other datasets will be very interesting. Moreover, it has to be considered if it is possible to adapt this algorithm in order to solve other machine learning problems, such as multiple classification or regression.

1. Introducció

1.1. Motivació i context

Dins el mon de l'aprenentatge automàtic (l'anomenat *Machine Learning*), s'han desenvolupat al llarg dels anys un gran nombre de tècniques i metodologies d'aprenentatge [1] [2], tals com els arbres de decisió (DT), les xarxes neuronals artificials (ANN) i els mètodes de kernel. Degut a la seva recent popularitat, s'ha incrementat el desenvolupament de tècniques cada cop més eficients per a realitzar les diverses tasques que aquesta disciplina presenta.

Dins dels mètodes de kernel, cal destacar la creixent popularitat de les SVMs o Support Vector Machines (Màquines de Vectors Suport) [3]. Les SVMs van ser dissenyades en un principi per a resoldre problemes de classificació binària, en la qual mostren excel·lents resultats de generalització [4]. És per aquest motiu que la recerca en aquesta tècnica ha estat quantiosa els últims anys, i s'han aconseguit grans millores de rendiment, tant de l'error de classificació com del temps d'execució.

Un dels principals avantatges de les SVMs és la seva gran versatilitat. Aquesta tècnica forma part dels mètodes de kernel [5], que implica que hi ha una part del mètode que és configurable amb una funció de kernel. Aquest kernel és un regulador directe de la complexitat del model, ja que aquesta depèn directament de la complexitat del kernel.

Un altre dels seus avantatges és la parametrització del problema. Totes les SVMs tenen un paràmetre de cost C que regula el grau d'ajustament del model a les dades. El paràmetre C pot tenir valors entre 0 i $+\infty$, i la seva tria incideix directament en la qualitat de la solució obtinguda. A més, alguns kernel (com el kernel RBF) tenen també paràmetres interns que s'han de configurar (γ o bé σ^2 per a RBF) i que també tenen gran incidència en la qualitat de la solució. Independentment, existeixen formulacions alternatives que utilitzen altres parametritzacions diferents al paràmetre de cost C [6].

Tot i les grans avantatges que presenta aquesta tècnica a nivell de classificació, el seu principal inconvenient és el seu elevat cost computacional. La resolució trivial del problema té cost temporal $O(n^3)$ i cost espacial $O(n^2)$ [7]. Tot i això, el temps d'execució empíric depèn de diversos factors, com la complexitat de les dades (separabilitat, nivell de soroll...) i dels valors de certs paràmetres (Un valor petit del paràmetre de cost C redueix significativament el temps d'entrenament de la SVM) [8].

És per aquest motiu que són de gran interès aquelles tècniques que tenen com a objectiu la reducció del cost de l'entrenament d'una SVM, tant espacial com temporal. Hi ha algunes tècniques que, tot i no reduir la complexitat asimptòtica del mètode, són capaces de reduir el temps de computació de tal manera que problemes que resulten intractables amb una SVM resulten perfectament resolubles. Cal remarcar, però, que no hi ha hagut una estandardització d'aquestes tècniques, i que només algunes d'elles apareixen implementades en les principals llibreries dels diversos llenguatges de programació.

1.2. Treballs previs

Actualment, no hi ha una gran varietat de mètodes dissenyats per a aconseguir una reducció del cost d'entrenament, i els que hi ha no han estat estandarditzats. Generalment, els resultats obtinguts amb aquests mètodes no tenen per què ser coincidents amb els de la SVM original, però tots tenen com a objectiu obtenir resultats raonables amb un cost significativament menor.

Per al problema de la reducció del cost d'entrenament d'una SVM, hi ha diverses solucions més o menys satisfactòries. Aquestes solucions les podem dividir en dos tipus:

1. Aquelles solucions que modifiquen el funcionament intern de la SVM per a optimitzar-ne el rendiment.

2. Aquelles solucions que modifiquen les dades per a que una SVM estàndard pugui veure accelerat el seu rendiment.

Les solucions del primer conjunt tenen com a principal finalitat l'acceleració de l'entrenament sense pèrdua de rendiment per a la classificació, mentre que les solucions del segon conjunt tenen com a finalitat una reducció de les dades d'entrada, de manera que s'aconsegueixi la màxima reducció amb el mínim de pèrdua de rendiment.

Dins del primer grup cal destacar les tècniques de *Chunking* [9] i, amb major importància, el mètode de SMO (*Sequential Minimal Optimization*) [10], un dels més utilitzats per a accelerar l'entrenament de la SVM. Aquest últim aconsegueix reduir el cost temporal a $O(n^2)$ i el cost espacial a $O(n)$, i és pràcticament la única de les tècniques que ha estat estandarditzada (utilitzada per la majoria de llibreries disponibles).

Altres mètodes presenten directament una variant de les SVM, com les CVM (*Core Vector Machines*) [11], les RSVM (*Reduced Support Vector Machines*) [12] [13] o les WSVM (*Weighted Support Vector Machines*) [14].

Dins el segon grup, actualment no hi ha una gran varietat de tècniques. Les primeres idees es basen en *Reduced Sets* [15] [16], obtinguts generalment mitjançant combinacions lineals de les dades i càlculs de pre–imatges. La idea és obtenir un nou conjunt de dades que representi el millor possible les dades originals.

Hi ha un conjunt de tècniques dissenyades per identificar aquelles dades "rellevants" per al rendiment de la SVM, i utilitzar-les per a l'entrenament. Així, el conjunt de dades rellevants serà aquell amb una major "proximitat" al conjunt de dades de la classe contraria.

Una idea basada en aquest concepte és la de la tècnica de *Opposite Maps* [17]. Aquesta tècnica identifica com a dades rellevants aquelles més properes a un conjunt de representants de la classe contrària, seleccionats mitjançant algun algoritme de clustering.

Un altre idea és el de l'anomenada *Risk Zone* [18] i la seva extensió, la *Generalized Risk Zone* [19]. La RZ determina, per a dues classes, les dades que tenen un risc més gran de ser classificades incorrectament, en base a diverses mesures de distància respecte un representant de la classe contraria. La GRZ, per contra, tracta cada conjunt de dades com una distribució de probabilitat, i no com un únic representant. D'aquesta manera, la proximitat al conjunt de dades deriva del valor de la divergència de Cauchy–Schwartz [20] entre distribucions de probabilitat. Així, el resultat és molt més flexible i precís per a estructures complexes que la RZ.

1.3. Objectius del projecte

En aquesta secció s'especificaran els objectius del projecte. Primer de tot distingirem entre els possibles objectius que es podrien plantejar. A continuació, es justificarà la tria dels objectius realitzada, i es procedirà a una especificació més detallada.

En base a l'anàlisi dels treballs previs, podem concloure que el primer conjunt de tècniques (aquelles que modifiquen el funcionament intern de la SVM) ja es troben relativament desenvolupades, essent SMO el seu màxim exponent. Per contra, al segon conjunt de tècniques (aquelles que modifiquen les dades i apliquen una SVM estàndard) no hi ha una gran varietat mètodes disponibles, i la seva eficàcia no és massa gran.

És per aquest motiu que aquest projecte estarà enfocat principalment en dissenyar un mètode de la segona família, que millori el rendiment de les tècniques ja existents i ho faci de manera eficient. A més, és important que el mètode tingui una forta base teòrica, ja que si té un bon funcionament se n'han de poder justificar les causes.

Així doncs, l'objectiu principal del projecte és el disseny d'un algoritme que realitzi un **filtratge** de les dades de tal manera que la SVM pugui treballar directament amb les dades filtrades. A més, l'algoritme ha de permetre generar també un **model predictiu** que permeti realitzar la tasca de predicció de manera independent a l'aplicació de la SVM. Aquest algoritme ha de tenir en compte les següents limitacions:

1. El seu cost espacial i temporal ha de ser regulable per l'usuari mitjançant un conjunt de paràmetres de configuració. Amb la configuració adequada, s'ha de poder assolir un cost lineal respecte l'entrada, tant temporal com espacial.

2. S'ha de poder aplicar al mateix tipus de dades d'entrada que una SVM accepti.

3. El temps d'execució del mètode ha de ser significativament menor que el temps d'execució d'una SVM sobre les mateixes dades.

4. El temps d'execució d'una SVM amb les dades filtrades ha de ser significativament menor que el de la mateixa SVM amb les dades sense filtrar.

5. El rendiment d'una SVM amb les dades filtrades ha de ser el més semblant possible al de la mateixa SVM amb les dades sense filtrar.

Tot i que les SVM es poden utilitzar per a diferents tasques, tals com la classificació (binària o múltiple) o la regressió, en aquest projecte ens centrarem principalment en el problema de la classificació binària. Si el temps ho permet, però, es valorarà la possibilitat d'estendre l'algoritme dissenyat a problemes de classificació múltiple i de regressió.

1.4. Desglossament dels objectius

Els objectius definits a l'apartat anterior són relativament genèrics, degut a la gran flexibilitat del projecte. Aquests objectius es concretaran en aquest apartat, enumerant-los de manera que en puguem fer una futura avaluació en finalitzar el projecte.

Així doncs, els objectius del projecte són els següents:

1. Dissenyar un (o diversos) **algoritmes eficients** que permetin realitzar una tria de les dades d'entrada, de manera que es pugui entrenar una SVM (per a classificació binària) amb aquestes dades i obtenir resultats el més semblants possibles als d'entrenar la mateixa SVM amb les dades originals.

2. Generar un **model predictiu** de manera eficient obtingut a partir del mateixos mètodes utilitzats per al filtratge de les dades.

3. Obtenir un **model probabilístic** a partir dels models predictius generats.

4. Validar **experimentalment** els diversos algoritmes dissenyats, i triar-ne aquell que mostri un millor rendiment, tant a nivell de cost espacial/temporal com a nivell de resultats.

5. Comparar l'algoritme triat amb **altres mètodes** que realitzin una tasca semblant, i **millorar-ne** els resultats.

6. Estendre l'algoritme dissenyat per a resoldre problemes de **classificació múltiple**.

7. Estendre l'algoritme dissenyat per a resoldre problemes de **regressió**.

Els objectius proposats seran validats en finalitzar el projecte, tot comentant els que s'hagin assolit i justificant les causes que han impedit l'assoliment d'aquells casos en que no hagi estat possible.

1.5. Organització del document

Des d'aquest punt en endavant, el document seguirà l'estructura següent:

2. **Planificació del projecte**: En aquest apartat es recull tot aquell treball previ a l'inici formal del projecte, com la planificació temporal, l'estipulació d'una metodologia de treball i la gestió econòmica, social i ambiental del projecte.

3. **Estat de l'art**: En aquest apartat es realitzarà una síntesi de tots els coneixements previs sobre mètodes kernel i màquines de vectors suport (SVM) necessaris per a la comprensió d'aquest projecte. S'analitzarà també el funcionament de la versió kernelitzada de FDA, KFDA. També en aquest apartat s'analitzaran a fons aquelles tècniques que tenen una major similitud amb els nostres objectius, i en quina mesura aconsegueixen satisfer-los.

4. **Formalització del problema**: En aquest apartat es formalitzarà el problema a resoldre, s'explicaran aquells conceptes específics dels algoritmes dissenyats i s'especificarà la notació a utilitzar en la seva consegüent explicació.

5. **Algoritmes dissenyats**: En aquest apartat es realitzarà una descripció formal dels algoritmes dissenyats, acompanyada sempre que sigui possible d'esquemes i pseudocodi que en faciliti la comprensió. Es justificarà la base teòrica de les decisions preses als diversos algoritmes i es realitzarà un càlcul teòric dels seus costos asimptòtics. A més, s'intentarà justificar quin dels algoritmes és el millor, en base als seus costos asimptòtics i al seu rendiment esperat.

6. **Resultats experimentals**: En aquest apartat s'exposaran amb un gran nivell de detall els experiments als que es pretén sotmetre als algoritmes dissenyats. Aquests experiments inclouen un conjunt d'experiments amb casos sintètics (generats seguint alguna distribució de probabilitat fixada) i també un conjunt d'experiments amb casos reals, obtinguts de diversos conjunts de dades provinents de repositoris com UCI o Kaggle.

7. **Conclusions**: En aquest apartat es realitzarà un anàlisi de la feina realitzada. Es valorarà el grau d'assoliment de la planificació realitzada, així com el rigor amb el que s'ha seguit la metodologia establerta i el grau de coherència amb el que s'han estudiat les alternatives a l'algoritme triat. A més, s'analitzarà el conjunt de coneixements obtinguts al llarg dels estudis de grau que s'han pogut aplicar amb èxit en aquest projecte, i es revisarien els objectius inicialment proposats per a validar quins s'han pogut assolir i quins no.

2. Planificació del projecte

2.1. Planificació inicial

En aquesta secció s'exposarà l'estructura del projecte, les etapes en les que s'ha dividit i el temps necessari per a realitzar-les. El projecte va començar oficialment el 18 de Setembre de 2017, i finalitza amb l'entrega d'aquest document el 15 de Gener de 2018.

Per tal de facilitar la planificació temporal i no solapar les diferents tasques a realitzar, el projecte s'ha dividit en tres grans blocs:

- **Planificació del projecte**: Aquesta és la primera etapa a realitzar, i es desenvolupa sota la tutela dels professors de l'assignatura de GEP. Un cop finalitzada aquesta part, es podrà procedir a la realització del projecte en sí. Aquesta fase té una duració aproximada d'un mes, des del 18/09 fins al 23/10.

- **Execució del projecte**: En aquesta fase es realitza el gruix del treball: S'ha de realitzar la totalitat del disseny de l'algoritme o algoritmes a presentar, amb les seves corresponents implementacions i els resultats dels mateixos enfront de diversos jocs de probes. Aquesta fase té una duració aproximada d'un mes i mig, des del 24/10 fins al 18/12.

- **Redacció de la documentació**: En aquesta fase, cal documentar detalladament els algoritmes resultants de l'execució del projecte. A més cal recollir també en un document els resultats de les probes realitzades i un pseudocodi que representi adequadament els algoritmes dissenyats. Aquesta fase té una duració aproximada de dues setmanes, des del 19/12 fins al 31/12.

Tasca	Duració Aproximada
Planificació del projecte	90 h
Execució del projecte	360 h
Redacció de la documentació	50 h
Total	500 h

Taula 2.1: Distribució d'hores al projecte

En total, el cost estimat en hores de dedicació al projecte és d'unes 500 hores. Cal remarcar que la data límit per a realitzar l'entrega del projecte és el 15 de Gener de 2018. Així doncs, resta un període de 15 dies des de la finalització de la última fase del projecte. Aquests dies han estat reservats per si cal realitzar una ampliació d'alguna de les fases, o per a desenvolupar els objectius secundaris que ens hem plantejat.

2.2. Metodologia de treball

En aquesta secció s'exposarà la metodologia establerta per a l'execució d'aquest projecte. L'especificació de la mateixa és molt important per a evitar problemes amb la planificació temporal, així com garantir la detecció precoç de problemes durant el desenvolupament del projecte. Així doncs, és crític seguir aquesta metodologia de manera molt rigorosa per a garantir el bon funcionament del treball.

Degut a les característiques del projecte, resulta molt difícil aplicar una metodologia clàssica amb una planificació temporal estàndard. Així, la metodologia que es seguirà serà més semblant a la família de metodologies àgils que a les clàssiques.

La metodologia bàsica que s'agafarà com a referència és la metodologia *scrum*, ja que el seu mètode de cicles curts i prototipats ràpids s'ajusta a les nostres necessitats de detectar precoçment aquelles idees que no han de ser desenvolupades.

Aquesta metodologia, però, no es pot aplicar directament, ja que en un principi està dissenyada per a projectes amb un major nombre de participants. Com que aquest projecte és individual, tota la part d'interacció i comunicació serà omesa, però aprofitarem la part de divisió de tasques i de cicles curts de producció amb la finalitat de presentar un prototipus al final de cada cicle.

El disseny de la metodologia s'ha realitzat durant la fita inicial. Aquesta determina que la feina es realitza en iteracions de dos cicles, que funcionen de la manera següent:

1. Es realitza una cerca bibliogràfica de diversos mètodes actuals per a la resolució del problema.

2. Es dissenya un esbós d'un algoritme que realitzi la tasca desitjada.

3. Es divideix l'algoritme anterior en el major nombre de fases diferenciades possible.

4. S'analitza cada fase per separat, tot cercant bibliografia que ajudi a resoldre el problema concret que es troba. S'intenta realitzar una optimització de la fase, tant a nivell de temps d'execució com de qualitat de la solució trobada.

5. Es construeix un prototipus de l'algoritme i es valora el seu rendiment, tant a nivell de temps d'execució com de qualitat de la solució trobada.

6. Si els resultats no són satisfactoris, tornar al pas 4.

7. Un cop el mètode ha estat optimitzat el màxim possible, es valoren alternatives per a cadascuna de les fases i es torna al pas 4.

8. Un cop s'han explorat totes les alternatives viables per a les diverses fases de l'algoritme, es torna al pas 1 i es pensa algoritmes alternatius per a realitzar la feina.

9. Un cop s'han ideat suficients algoritmes i s'han optimitzat per fases, es realitza una comparació global d'entre totes les opcions i es valoren les més viables, eficients i amb menor cost temporal.

Lògicament, els passos 7 i 8 es poden repetir indefinidament, sense cap criteri estricte de satisfacció ni un límit en el nombre de possibilitats. Aquest criteri, per tant, s'haurà de determinar en funció del temps invertit, i del temps que resta del projecte.

2.3. Possibles problemes i solucions

Dins del desenvolupament del projecte, ens podem trobar un conjunt de problemes o contratemps diversos. En aquest apartat, els analitzarem i en raonarem algunes possibles solucions.

2.3.1. Mal disseny de l'algoritme inicial

Pot ser que l'algoritme ideat no resulti en una bona idea per una gran quantitat de motius: Per que té un cost temporal massa elevat, per que requereix una quantitat de memòria massa elevada, per que el rendiment del mètode no és prou bo...

En qualsevol cas, la solució és sempre intentar detectar aquests inconvenients el més aviat possible, tot optimitzant primer les parts més complexes per tal de determinar la complexitat mínima del mètode i valorar si cal descartar-lo o no.

2.3.2. L'algoritme ideat ja existeix

Un altre problema pot ser idear un algoritme que es creu nou, realitzar-ne la optimització i implementació, i adonar-se posteriorment que aquest mètode ja ha estat desenvolupat.

Això té una doble conseqüència negativa: Per una banda, s'ha invertit una certa quantitat de temps en desenvolupar un mètode que ja ha estat desenvolupat i documentat, mentre que d'altra banda no és la finalitat del nostre treball el realitzar un compendi de treballs anteriors, sino el disseny d'un nou mètode que en pugui millorar el rendiment.

Així, hem d'intentar evitar al màxim possible el disseny d'algoritmes sense una base bibliogràfica suficient, i hem d'assegurar-nos que la nostra idea no ha estat desenvolupada prèviament abans de realitzar la nostra optimització i implementació del mètode.

En cas que trobem que la nostra idea implementada resulti haver estat ja dissenyada anteriorment, no és necessari descartar-la completament. En comptes d'això, es pot decidir d'integrar la solució en una de nova, tot reciclant el que ja s'ha pensat per a produir un mètode nou.

2.3.3. Problemes de calendari

Un dels principals riscos d'aquest projecte és no disposar de temps suficient per a explorar totes les possibles idees que sorgeixen. És molt difícil acotar temporalment les diverses fases, ja que no totes les idees requereixen el mateix temps per a ser desenvolupades i/o implementades.

És important dedicar prou temps a totes les idees, però també és important identificar el més aviat possible aquelles idees que són massa complexes per a ser implementades o aquelles que no compliran els requisits especificats pel nostre problema.

2.3.4. Errors d'implementació

Un altre problema que ens podem trobar és el de trobar-nos amb errors en la implementació dels prototipus. En el millor dels casos, aquests errors només suposaran un cost temporal per al projecte mentre es cerquen els errors i s'arreglen.

Els errors d'implementació, però, poden dur a problemes més seriosos, com pot ser el fet de descartar certes idees degut a que es cregui que no funcionen bé, mentre que la realitat és que no funcionen bé degut a un error d'implementació.

És per aquest motiu que es prenen dues decisions respecte al desenvolupament del projecte. La primera és utilitzar per a la programació el llenguatge R, que a més d'implementar diverses llibreries d'estadística i Machine Learning, és un llenguatge de *scripting* que facilita la ràpida construcció de prototipus i la detecció d'errors més ràpidament, a més de permetre una major facilitat per a realitzar *debugging*.

A més, en cas que una implementació d'un mètode no resulti satisfactòria, es realitzarà una re-implementació del mètode completament independent de l'anterior, a fi de comprovar si efectivament és el mètode el que no funciona o és la implementació realitzada la que és la font del mal funcionament detectat.

2.4. Eines per al desenvolupament

En aquest apartat es llistarà el conjunt d'eines i tecnologies que es requeriran durant l'execució del projecte. La gran majoria d'elles són gratuïtes, sigui pel fet que es tracta de software gratuït o per que la universitat disposa de llicències que en permeten l'ús en l'àmbit acadèmic.

Principalment, s'utilitzarà per al desenvolupament del projecte el llenguatge de programació R, ja que s'adapta molt bé a les necessitats del nostre projecte. Tot i això, i com que interessa que el mètode pugui ser utilitzat per el màxim d'usuaris possible, es planteja la possibilitat de realitzar implementacions del mètode en altres llenguatges utilitzats comunament per a realitzar tasques de Machine Learning, com Matlab o Python.

A més, és necessari conservar tots els prototipus programats, tant si han funcionat adequadament com si no. Això és per tal de garantir que, si posteriorment sorgeix alguna idea que permet fer funcionar alguna idea anteriorment descartada, es pugui re-construir i adaptar un prototipus anterior a la nova idea.

La millor manera de realitzar aquesta tasca és utilitzar un repositori per a mantenir el codi. En aquest cas, s'utilitzarà un sistema basat en Git, acompanyat d'un servidor de repositoris online (GitHub o similar) que ens permeti conservar i compartir el nostre codi a internet.

Així, aconseguim una doble tasca: El nostre codi queda protegit de possibles eliminacions accidentals, pèrdua del dispositiu físic on s'emmagatzema o altres riscos, i a més el tutor del treball pot consultar en tot moment la feina de programació que s'està realitzant, i contribuir a la detecció precoç d'errors de programació.

2.5. Seguiment del projecte

En un projecte de les característiques d'un treball de fi de grau (TFG) és molt important que hi hagi una bona comunicació amb el director del projecte, a fi que es garanteixi que es segueix la planificació inicial de manera rigorosa i que el projecte no s'allunya innecessàriament dels objectius pre-establerts. La opinió del director, a més, és molt important de cara a totes les etapes del projecte, i és necessari que pugui realitzar una supervisió suficient del mateix.

Per a fer un seguiment general del projecte, es manté per tant amb el director una constant comunicació per correu electrònic i amb reunions presencials, per tal de consultar qualsevol nova idea que sorgeixi per a l'algoritme. Això és important, ja que el tutor pot valorar molt més eficientment si una idea val la pena ser explorada o no.

També serà compartit amb el tutor el repositori de Git del projecte, per tal que pugui consultar en tot moment l'estat dels prototipus del projecte i col·laborar, si vol, en la revisió dels mateixos.

A més, és primordial documentar per escrit cada possible mètode i les seves fases, ja que és important que quedi constància escrita de les idees que han sorgit. A més, cal acompanyar cada documentació d'un algoritme escrit en pseudocodi que reflecteixi les necessitats d'implementació del mètode i permeti un càlcul ràpid del seu cost espacial i temporal.

2.6. Gestió econòmica

En ser aquest projecte un treball de final de grau (TFG), és interessant realitzar un petit estudi de la viabilitat econòmica del projecte, com si fos un projecte d'un negoci real. Els costos econòmics del projecte es deriven de la planificació horària realitzada i de les eines utilitzades. El pressupost, per tant, ha de tenir en compte tots aquests aspectes del projecte.

2.6.1. Pressupost de Recursos Humans

Aquest projecte serà desenvolupat per una sola persona, per tant aquesta serà l'encarregada de realitzar totes les tasques assignades. El sou del dissenyador s'ha fixat en 10€/h, mentre que el de la persona que implementarà els algoritmes s'ha fixat en 7.5€/h. Així, podem dividir el cost segons les fases de la manera següent:

Fase	Dissenyador	Implementador	Temps per fase (h)	Cost per fase (€)
Planificació del projecte	90h	0h	90h	900€
Execució del projecte	300h	60h	360h	3.450€
Documentació del projecte	25h	25h	50h	438€
Temps per rol (h)	415h	85h	500h	
Cost per rol (€):	4.150€	638€		4.788€

Taula 2.2: Pressupost de Recursos Humans

2.6.2. Pressupost de Hardware

A l'hora de realitzar el projecte només serà necessari l'ús d'un ordinador portàtil. Per tant, els costos de hardware seran els següents:

Producte	Preu	Unitats	Vida útil (anys)	Amortització
Portàtil Acer Aspire	600€	1	5 anys	60€

Taula 2.3: Pressupost de Hardware

2.6.3. Pressupost de Software

A l'hora de realitzar el projecte es requeriran de diversos entorns software. Per tant, els costos de software seran els següents:

Producte	Preu	Unitats	Vida útil (anys)	Amortització
Windows 8.1 Pro	0€	1	3 anys	0€
Ubuntu 16.04 LTS	0€	1	3 anys	0€
Microsoft Office 2016	69€	1	1 any	0€
RStudio	0€	1	–	0€
Python3	0€	1	–	0€
Matlab	0€	1	–	0€

Taula 2.4: Pressupost de Software

Cal destacar que la el cost de la llicència de Windows va inclosa en el preu del portàtil, de manera que no té cap cost com a software independent. A més, cal remarcar que el treball en Matlab es realitzarà amb les llicències disponibles per als estudiants de la UPC.

2.6.4. Altres despeses

Les altres despeses habituals (llum, connexió a internet, paper...) no es comptabilitzaran, ja que el projecte es realitzarà en una llar particular o al recinte de la universitat, sense que la presencia d'aquest projecte suposi un cost addicional d'aquestes despeses en aquests llocs.

2.6.5. Pressupost total

En global, el pressupost estimat per a aquest projecte és el següent:

Concepte	Pressupost
Recursos Humans	4.788€
Hardware	60€
Software	0€
Altres despeses	0€
Total:	4.848€

Taula 2.5: Pressupost total

2.6.6. Viabilitat econòmica

Per a estudiar la viabilitat econòmica d'aquest projecte, cal destacar que tant els costos de software com els de hardware ja han estat realitzats i amortitzats, de manera que no suposen realment una despesa pel projecte. Per tant, la totalitat dels costos del projecte seran aquells derivats dels recursos humans.

En global, s'estima que el projecte tindrà un cost de casi 5.000 €. Aquest projecte, per tant, és perfectament viable i es podrà realitzar sense problemes.

2.7. Sostenibilitat i compromís social

Avui en dia, la sostenibilitat en el desenvolupament de projectes és un factor amb una importància creixent. És important comptabilitzar el consum de recursos que aquest projecte pot comportar, ja que és un aspecte a tenir molt en compte a la societat actual. En aquest projecte, s'ha analitzat la sostenibilitat mitjançant tres aspectes diferents: la seva dimensió econòmica, social i ambiental.

2.7.1. Sostenibilitat econòmica

El projecte és sostenible econòmicament, ja que els costos en general són molt reduïts i ho poden ser encara més. Això és degut a la naturalesa del projecte, que pot finalitzar anticipadament (reduint així els costos de recursos humans) si es troba un algoritme adient de manera ràpida. Aquest projecte, per tant, es pot escurçar però no es pot allargar, fet que en garanteix la sostenibilitat econòmica.

2.7.2. Sostenibilitat social

En ser aquest projecte un projecte de recerca teòric, les seves implicacions socials es limiten a aquelles que ja tenen les SVM, que no són diferents de les que realitza qualsevol sistema d'aprenentatge automàtic. Igualment, tampoc es pot afirmar que aquestes conseqüències puguin provenir del nostre projecte, ja que aquest només realitzaria una acceleració d'aquestes tècniques, però no en realitzaria la funció. Per tant, aquest projecte no té cap vessant social que s'hagi de mencionar.

2.7.3. Sostenibilitat ambiental

Aquest projecte no té cap mena de repercussió ambiental, exceptuant el paper consumit en els esborranys del disseny de l'algoritme. En ser un projecte bàsicament teòric, la gran majoria del temps tampoc s'utilitzarà l'ordinador, de manera que el consum elèctric serà molt reduït. Així, aquest projecte no té cap vessant ambiental que s'hagi de mencionar.

2.7.4. Matriu de sostenibilitat

A partir de les informacions especificades als apartats anteriors, podem quantificar les caselles de la **matriu de sostenibilitat**. Aquesta reflexa de manera ràpida i clara la sostenibilitat del nostre projecte, i permet fer-se una idea de les implicacions del treball.

Dimensió	PPP	Vida útil	Riscs
Econòmica	Factura	Pla de viabilitat	Riscs econòmics
	10/10	18/20	0/–20
Social	Impacte personal	Impacte social	Riscs socials
	8/10	12/20	0/–20
Ambiental	Consum del disseny	Petjada ecològica	Riscs ambientals
	9/10	18/20	–1/–20
Rang de sostenibilitat	27/30	48/60	–1/–60
	74/90		

Taula 2.6: Matriu de sostenibilitat

2.8. Identificació de lleis i regulacions

En ser aquest un projecte d'investigació, cal destacar que el resultat del projecte no serà un producte amb aplicació directa al mercat, sinó que altres productes podran utilitzar els nostres resultats, i seran aquests els que hauran de considerar les respectives legislacions i regulacions. El nostre projecte, per tant, no es troba regulat per cap legislació.

En qualsevol cas, només pel fet de ser un Treball de Fi de Grau (TFG) associat a la Facultat d'Informàtica de Barcelona (FIB), aquest projecte es troba subjecte a la normativa de la UPC i de la facultat sobre els Treballs de Fi de Grau.

3. Estat de l'art

3.1. Mètodes de kernel

3.1.1. Classificació en espais vectorials i espais de Hilbert

Un dels problemes fonamentals de la teoria d'aprenentatge és el de la **classificació binària**: Suposem que tenim objectes de dues classes. Aleshores, apareix un nou objecte, i hem de decidir a quina de les dues classes conegudes pertany. Formalment, disposem d'un conjunt de dades empíriques:

$$(x_1, t_1), (x_2, t_2), \dots, (x_n, t_n) \in D \times \{\pm 1\}$$

On *D* és el conjunt de totes les possibles dades empíriques, anomenat **domini**. Cadascuna de les x_i s'anomenen **instàncies**, **casos** o **observacions**, i les t_i s'anomenen **etiquetes**, ***targets*** o **classes**. En general, definim $X = \{x_1, x_2, \dots, x_n\}$ com el conjunt que conté els elements de la nostra mostra, i $T = \{t_1, t_2, \dots, t_n\}$ com el conjunt de tots els *targets*. L'objectiu de la classificació binària és definir una funció de la forma:

$$f: D \to \{\pm 1\}$$

Que realitzi una **classificació** el més correcta possible dels nous objectes que apareixen. Aquesta funció ha d'intentar reduir al màxim l'error realitzat en la classificació. Aquest es defineix de la manera següent:

$$err(x, t, f) = \frac{1}{2}|f(x) - t|$$

Així, l'error és 0 si la predicció és correcta, i 1 si no ho és. Si realitzem una partició del conjunt X en dos conjunts X_{train} (d'entrenament) i X_{test} (de test), podem entrenar un classificador amb el primer conjunt i validar el seu rendiment mesurant l'error en els elements del segon conjunt. A l'error obtingut amb el conjunt d'entrenament se l'anomena **error d'entrenament**, mentre que a l'obtingut amb el conjunt de test se l'anomena **error de test** o **error generalitzat**. És el segon el que ens indica realment com de bé realitza el nostre classificador la seva tasca. Cal destacar que tenir un error d'entrenament baix (o nul) no és garantia de que la funció tingui un error generalitzat baix, ja que aquest classificador pot tenir un ajust massa gran a les dades d'entrenament.

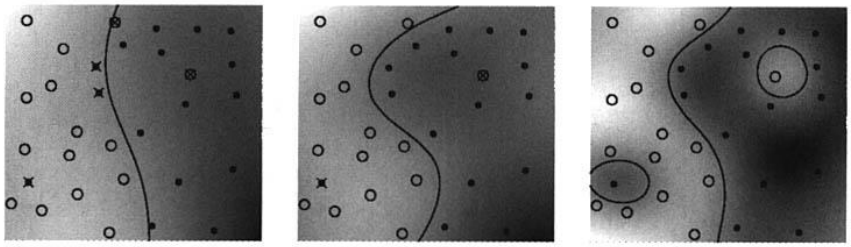

Figura 3.1: Classificadors de diferent complexitat [5]

Per exemple, en els classificadors de la figura 3.1, podem observar que el classificador de la dreta del tot té una complexitat molt elevada, fet que pot dur a classificar malament alguns punts propers a aquells punts aïllats. En canvi, el classificador de l'esquerra és massa senzill, i no pot realitzar adequadament la tasca. Així, el classificador del mig és el que troba el millor equilibri entre la complexitat del classificador i l'error d'entrenament.

Generalment, cada instància que apareix es codifica com un vector en un cert espai vectorial $\mathbb{R}^d$, de dimensió dependent del nombre d'atributs de les instàncies. D'aquesta manera, podem establir un concepte de **similitud** entre diversos casos, i intentar predir la classe d'una nova observació en funció de la classe d'altres observacions similars.

És precisament la definició d'aquest concepte de similaritat el que distingeix els diversos mètodes d'aprenentatge automàtic. La gran majoria d'ells utilitzen les eines que aporta l'àlgebra lineal per a realitzar aquesta tasca, com el càlcul de distàncies, angles, vectors directors i projeccions.

Aquestes tècniques, però, tenen la limitació de dependre d'una representació a $\mathbb{R}^d$ de les dades. En realitat, tots els càlculs anteriors es poden realitzar a partir únicament del **producte escalar** de les dades. Aquest es defineix com:

$$\langle x_1, x_2 \rangle = \sum_{i=1}^{d} [x_1]_i \cdot [x_2]_i$$

On $[x]_i$ correspon al *i*-èssim element del vector *x*. Aquest producte es pot representar de diverses maneres:

$$\langle x_1, x_2 \rangle = x_1 \cdot x_2 = x_1^T x_2$$

Un cop definit el producte escalar de les dades, és possible realitzar la resta de càlculs d'un espai vectorial només en funció d'aquest producte.

Per exemple, la **longitud** (o **norma**) d'un vector es pot calcular com:

$$\|x\| = \sqrt{\langle x, x \rangle}$$

La **distància** entre dos punts es pot calcular com:

$$d(x_1, x_2) = \|x_1 - x_2\| = \sqrt{\langle x_1, x_1 \rangle + \langle x_2, x_2 \rangle - 2\langle x_1, x_2 \rangle}$$

El **cosinus de l'angle** entre dos vectors es pot calcular com:

$$\cos \angle(x_1, x_2) = \frac{\langle x_1, x_2 \rangle}{\sqrt{\langle x_1, x_1 \rangle}\sqrt{\langle x_2, x_2 \rangle}}$$

En general, però, aquests càlculs requereixen que el domini tingui definit un producte escalar. Si el domini és un espai vectorial $\mathbb{R}^d$, aquest tindrà definit el seu producte escalar estàndard, però un domini genèric no té per que tenir-lo definit.

En els casos en que el nostre domini no té definit el producte escalar, podem realitzar una transformació de les dades aplicant una funció que transformi cada dada a una representació en un espai on el producte escalar estigui definit. Fins i tot si el nostre domini és un espai vectorial $\mathbb{R}^d$, o és un espai amb el producte escalar definit, ens pot interessar aplicar una funció d'aquest estil per tal d'aconseguir transformacions no lineals de les dades abans del seu processat.

La funció de transformació s'anomena **Feature Map**. Aquesta és una funció de la forma:

$$\phi: D \to H$$

On l'espai H s'anomena **Feature Space**. Com a Feature Space s'utilitza un **espai de Hilbert**, que és un espai vectorial on hi ha definit el producte escalar i és complet respecte la norma induïda per aquest producte escalar. Tot espai de Hilbert compleix, a més, les característiques següents:

1) El seu producte escalar és **simètric**:

$$\langle x_1, x_2 \rangle = \langle x_2, x_1 \rangle$$

2) El seu producte escalar és **lineal**:

$$\langle a x_1 + b x'_1, x_2 \rangle = a\langle x_1, x_2 \rangle + b\langle x'_1, x_2 \rangle$$

3) El producte escalar d'un element amb si mateix és **positiu semi-definit**:

$$\langle x, x \rangle \geq 0$$

   On la igualtat es compleix únicament quan x és l'element nul.

4) Es compleix la **desigualtat triangular**:

$$d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$$

5) Es compleix la **desigualtat de Cauchy-Schwarz**:

$$\langle x_1, x_2 \rangle \leq \|x_1\| \cdot \|x_2\|$$

Per tant, si tenim un producte escalar definit que compleix aquestes condicions, podem utilitzar-lo per a calcular totes les relacions que hem definit anteriorment, fins i tot si el domini original no admet el producte escalar o no és un espai vectorial.

Així, es poden estendre els principals algoritmes d'aprenentatge automàtic a tot tipus de dades, mitjançant la definició d'un **Feature Map** adequat. A més, en aquelles dades amb una relació molt complexa, podem definir un Feature Map que en permeti un millor anàlisi i classificació.

3.1.2. Funcions de Kernel

Seguint la idea de l'apartat anterior, podem definir la **similitud** de dos elements del domini de la manera següent:

$$k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$$

On *k* s'anomena **kernel** o **funció de kernel** [21]. Aquesta es defineix:

$$k: D \times D \to \mathbb{R}$$

La funció de kernel es pot calcular explícitament si tenim definit el Feature Map. Per exemple, si definim un Feature Map:

$$\phi: \mathbb{R}^2 \to \mathbb{R}^4 \qquad \phi\begin{pmatrix}[x]_1\\[x]_2\end{pmatrix} = \begin{pmatrix}[x]_1^2\\[x]_2^2\\[x]_1[x]_2\\[x]_2[x]_1\end{pmatrix}$$

Aleshores, podem computar el valor de la funció de kernel:

$$k(x_1, x_2) = ([x_1]_1[x_2]_1 + [x_1]_2[x_2]_2)^2$$

Aquesta expressió es pot escriure en termes del producte escalar en el domini:

$$k(x_1, x_2) = \langle x_1, x_2 \rangle^2$$

Així, el producte escalar en l'espai de Hilbert es pot computar de manera senzilla a partir del producte escalar al domini i de la funció de kernel. Cal remarcar que tots els Feature Maps tenen associada una funció de kernel, però que no totes les funcions de dues variables són funcions de kernel.

Es defineix com a **matriu de Gram** o **matriu de Kernel** [21] la matriu:

$$K_{i,j} = k(x_i, x_j)$$

Aquesta matriu es construeix a partir de les dades de cada domini i té mida $n \times n$. Una matriu *K* es **semi-definida positiva** si, per a tot $c \in \mathbb{R}^n$, compleix:

$$c^T K c \geq 0$$

Es pot demostrar que, una funció $k: D \times D \rightarrow \mathbb{R}$ correspon a un producte escalar en algun espai de Hilbert si i només si la matriu de kernel generada per aquesta funció és semi-definida positiva [21].

El fet que una matriu sigui semi-definida positiva implica, per una banda, que els elements de la diagonal són positius (o 0); i per altra, que la matriu és simètrica. Així, la funció de kernel ha de complir també:

$$k(x, x) \geq 0$$
$$k(x_1, x_2) = k(x_2, x_1)$$

3.1.3. Construcció de Kernels

Generalment, és interessant disposar d'una gran varietat de funcions de kernel, ja que els rendiment dels diversos algoritmes que l'utilitzen en pot dependre de la tria. En general, hi ha tres maneres de construir un kernel [5]:

1) Construir explícitament un Feature Map, i calculant-ne l'expressió del seu producte escalar.

2) Construir una matriu simètrica semi-definida positiva a partir de les dades i utilitzar-la com a matriu de kernel.

3) Construir un nou kernel a partir de combinacions de kernels anteriors.

La manera més útil de construir noves funcions de kernel és el tercer mètode. Així, disposem d'una llista de regles per a la construcció de nous kernels: Siguin k_1, k_2 dos kernels vàlids, $c > 0$ una constant, f una funció i p un polinomi de coeficients positius. Així, son vàlids els següents kernels:

1) $k(x_1, x_2) = c \cdot k_1(x_1, x_2)$

2) $k(x_1, x_2) = f(x_1) k_1(x_1, x_2) f(x_2)$

3) $k(x_1, x_2) = p[k_1(x_1, x_2)]$

4) $k(x_1, x_2) = \exp\big(k_1(x_1, x_2)\big)$

5) $k(x_1, x_2) = k_1(x_1, x_2) + k_2(x_1, x_2)$

6) $k(x_1, x_2) = k_1(x_1, x_2) \cdot k_2(x_1, x_2)$

Amb aquest conjunt de propietats, podem construir el conjunt de kernels més estàndard. Aquests són els següents:

- Kernel lineal: $k(x_1, x_2) = \langle x_1, x_2 \rangle$

- Kernel polinòmic: $k(x_1, x_2) = (\langle x_1, x_2 \rangle)^d$
  - $d \in \mathbb{N}$

- Kernel polinòmic inhomogeni: $k(x_1, x_2) = (a \cdot \langle x_1, x_2 \rangle + c)^d$
  - $d \in \mathbb{N}$
  - $a > 0$
  - $c \geq 0$

- Kernel RBF: $k(x_1, x_2) = e^{\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)}$ o $k(x_1, x_2) = e^{(-\gamma \|x_1 - x_2\|^2)}$
  - $\sigma > 0$
  - $\gamma > 0$

- Kernel sigmoide: $k(x_1, x_2) = \tanh(a \cdot \langle x_1, x_2 \rangle - c)$
  - $a > 0$
  - $c > 0$

3.1.4. Dimensió de l'espai de Hilbert

Sigui *D* un conjunt no buit i $\phi: D \to H$ un Feature Map que genera una funció de kernel $k: D \times D \to \mathbb{R}$, la dimensió de l'espai de Hilbert *H* depèn explícitament del Feature Map utilitzat [22]. En aquelles construccions en que el kernel es construeix sense definició explícita, però, no és trivial determinar la dimensió de l'espai *H* construït.

- Per al kernel lineal, la dimensió de l'espai *H* és la mateixa que la del domini *D*.

- Per al kernel polinòmic, la dimensió de l'espai *H* és el nombre de monomis diferents d'ordre *d* que es poden construir amb els elements de *x*.

- Per al kernel polinòmic inhomogeni, la dimensió de l'espai *H* és el nombre de monomis diferents d'ordre *d* o menor que es poden construir amb els elements de *x*.

- Per al kernel RBF, la dimensió de l'espai *H* és infinita.

Tot i que la dimensió de l'espai *H* pot ser molt elevada, o fins i tot infinita, en la majoria de casos no s'utilitza la totalitat d'aquest espai per a realitzar la classificació. Concretament, si s'utilitza un mètode lineal, tots els vectors que apareguin al classificador seran, en realitat, combinacions lineals de les transformacions de les dades.

Així doncs, tots els vectors involucrats en un mètode lineal seran de la forma:

$$w = \sum_{i=1}^{n} \alpha_i \phi(x_i)$$

Amb $\alpha_i \in \mathbb{R}$. A més, podem definir un vector a partir d'una base de vectors formada pel conjunt de dades del domini. Així, el vector *w* el podem definir com:

$$\alpha = \begin{pmatrix} \alpha_1 \\ \dots \\ \alpha_n \end{pmatrix}$$

Tots els vectors amb els que treballarem es podran generar a partir de la base $B = \{\phi(x_1), \dots, \phi(x_n)\}$. Com que $K = B^T B$, el rang de *K* determina el rang de *B* i, per tant, la dimensió del subespai generat per les dades del domini.

Així doncs, tots els vectors obtinguts en algun punt d'un mètode lineal estaran continguts dins d'un subespai de dimensió (com a molt) *n*, contingut dins d'un espai de Hilbert de dimensió possiblement major (fins i tot, infinita). Si la dimensió de l'espai de Hilbert és menor que el nombre de dades, aquesta limita la dimensió del subespai.

3.1.5. Kernel Trick

Un dels inconvenients d'utilitzar mètodes de kernel és que el fet de treballar amb un Feature Map complica una mica el desenvolupament teòric dels mètodes. Aquest problema es resol molt fàcilment utilitzant l'anomenat *Kernel Trick* [5], que consisteix en re-definir totes les etapes del mètode de manera que les dades del domini només apareguin en forma de producte escalar $x_i^T x_j$.

Un cop modificades les aparicions de les dades, substituïm totes aquestes ocurrències per expressions de la forma $\phi(x_i)^T \phi(x_j)$. Aquestes últimes es poden computar per $\phi(x_i)^T \phi(x_j) = k(x_i, x_j)$, realitzant de manera efectiva una **kernelització** del mètode.

Utilitzant aquest concepte, podem tornar a realitzar tots els càlculs d'un espai vectorial a partir de l'expansió del vector i la matriu de kernel. Sigui α l'expansió del vector *w* i β l'expansió del vector *v*, tenim:

$$\|w\| = \sqrt{\alpha^T K \alpha}$$

$$d(w, v) = \|w - v\| = \sqrt{\alpha^T K \alpha + \beta^T K \beta - 2\alpha^T K \beta}$$

$$\cos \angle(w, v) = \frac{\alpha^T K \beta}{\sqrt{\alpha^T K \alpha}\sqrt{\beta^T K \beta}}$$

3.2. Màquines de Vectors Suport (SVM)

3.2.1. Hiperplans classificadors

Tot hiperplà a un espai vectorial es pot definir de la següent manera:

$$\{x \in \mathbb{R}^d | \langle w, x \rangle + w_0 = 0\} \text{ amb } w \in \mathbb{R}^d, w_0 \in \mathbb{R}$$

Així, un hiperplà es pot definir de dues maneres: A partir de la tupla $\pi: (w, w_0)$ o a partir de l'equació $\pi: \langle w, x \rangle + w_0 = 0$. D'aquesta manera, *w* correspon a un vector ortogonal a l'hiperplà que volem definir, i w_0 en determina la distància a l'origen.

En tot moment es pot calcular la distància (amb signe) d'un punt a l'hiperplà:

$$d(x_i, \pi) = t_i \frac{\langle w, x_i \rangle + w_0}{\|w\|}$$

Cal destacar que el signe d'aquesta distància depèn de la posició del punt a l'espai i de la seva classe. Així, considerarem que un punt es troba a distància positiva només si és de la classe positiva i es troba per sobre del pla, o si és de la classe negativa i es troba per sota. De la mateixa manera, un punt es trobarà a distància negativa si es troba situat a la banda contrària de l'hiperplà.

A més, cal remarcar que existeix una infinitat de tuples $\pi: (w, w_0)$ que generen exactament el mateix hiperplà. Concretament, si realitzem un escalat de l'hiperplà $\pi': \left(\frac{w}{c}, \frac{w_0}{c}\right)$, obtenim $\pi': \langle \frac{w}{c}, x \rangle + \frac{w_0}{c} = 0$, que és equivalent a $\pi: \langle w, x \rangle + w_0 = 0$.

Definim com a **hiperplà canònic** respecte d'un conjunt $x_1, x_2, [\dots], x_n \in H$ aquell hiperplà que compleix $\min_{1 \le i \le n} t_i(\langle w, x_i \rangle + w_0) = 1$. En aquest cas, es compleix:

$$\min_{1 \le i \le n} d(x_i, \pi) = \frac{1}{\|w\|}$$

Un cop definit el hiperplà de la manera més convenient per al problema, es pot utilitzar per a la classificació. Amb aquesta finalitat, definim una **funció de classificació** $f: H \to \{\pm 1\}$ associada a un hiperplà $\pi: (w, w_0)$,que assigna a cada element de l'espai la predicció de la seva classe. En aquest cas, la predicció es basa en determinar si el punt donat es troba per sobre o per sota de l'hiperplà classificador:

$$f(x) = \text{sgn}(\langle w, x \rangle + w_0)$$

Cal destacar, doncs, que dos hiperplans $\pi: (w, w_0)$ i $\pi': (-w, -w_0)$ corresponen al mateix hiperplà, però donen lloc a funcions de decisió diferents.

3.2.2. Marges geomètrics

Suposem que tenim un conjunt de dades $(x_1, t_1), (x_2, t_2), [\dots], (x_n, t_n) \in D \times \{\pm 1\}$ i un hiperplà π: (w, w_0) amb una funció de classificació $f(x) = sgn(\langle w, x \rangle + w_0)$ associada. El nostre objectiu és que π: (w, w_0) sigui tal que es compleixi $f(x_i) = t_i$ per a tot $1 \leq i \leq n$, és a dir, que realitzi una classificació correcta en tots els casos.

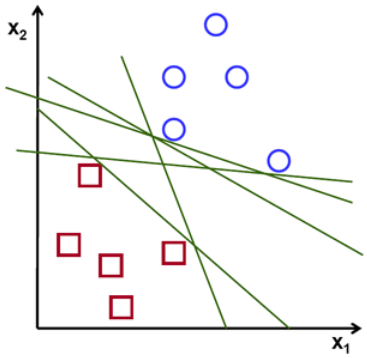

Figura 3.2: Diversos hiperplans que classifiquen correctament les dades [1]

Tot i això, hi pot haver diversos hiperplans que compleixin amb aquesta condició. Per exemple, a la figura 3.2 podem veure diversos hiperplans a $\mathbb{R}^2$ que classifiquen perfectament els dos conjunts de dades.

Per tal de decidir quin serà el millor d'ells, hem de definir el concepte de **marge**. Primer de tot, definim el **marge geomètric respecte d'un punt** x_i d'un hiperplà π: (w, w_0) com:

$$\rho_{\pi,i} = t_i \frac{\langle w, x_i \rangle + w_0}{\|w\|}$$

Ara, podem definir el **marge geomètric** d'un hiperplà π: (w, w_0) com:

$$\rho_{\pi} = \min_{1 \leq i \leq n} t_i \frac{\langle w, x_i \rangle + w_0}{\|w\|}$$

Així doncs, el marge d'un hiperplà respecte d'un conjunt de punts $x_1, x_2, [\dots], x_n \in H$ és la distància al més proper d'aquests punts.

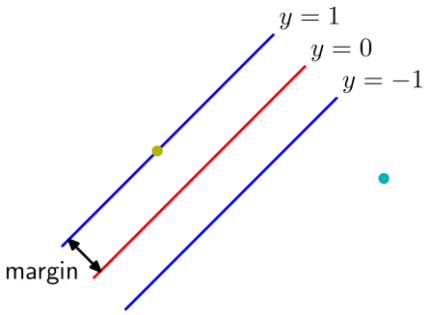

Figura 3.3: Marge d'un hiperplà [1]

Cal recordar que el marge d'un hiperplà amb un punt contingut a dins seu és 0. A més, si utilitzem un hiperplà canònic, el marge es computa de la manera següent:

$$\rho_\pi = \frac{1}{\|w\|}$$

Així doncs, resulta molt avantatjós utilitzar hiperplans canònics, ja que el mòdul del vector director té una implicació geomètrica directa.

3.2.3. Dimensió VC i maximitzacions del marge

Sigui $F = \{y: \mathbb{R}^d \to \{\pm 1\} | y = f(x)\}$ el conjunt de tots els possibles classificadors binaris i $(x_1, t_1), (x_2, t_2), [\dots], (x_n, t_n) \in D \times \{\pm 1\}$ un conjunt de dades a classificar, direm que un classificador $y \in F$ trenca *D* si classifica correctament $x_1, x_2, \dots, x_n$ per a les 2^n possibles combinacions de *targets*.

S'anomena **Dimensió VC** [22] d'un conjunt de classificadors *F* el major nombre *n* per al qual existeixen $x_1, x_2, \dots, x_n$ tals que existeix $y \in F$ que les trenca.

Per exemple, els classificadors lineals de dimensió *d* són aquells de la forma:

$$F_L^d = \{f(x) = \text{sgn}(\langle w, x \rangle + w_0) | x, w \in \mathbb{R}^d\}$$

La dimensió VC d'aquests classificadors es pot calcular, i resulta:

$$\text{dimVC}\left(F_L^d\right) = d + 1$$

Considerem ara una classe de models *F* amb $\text{dimVC}(F) = h$. Si entrenem un model de *F* amb un conjunt de dades *D*, i obtenim un error d'entrenament E_D i un error generalitzat *E* (l'error obtingut en classificar dades noves), amb probabilitat $1 - \mu$ tenim que:

$$E \leq E_D + H(h, \mu, n)$$

On es defineix:

$$H(h, \mu, n) = \sqrt{\frac{1}{n}\left(h\left(\ln\frac{2n}{h} + 1\right) + \ln\frac{4}{\mu}\right)}$$

Si el conjunt *D* és separable linealment, aleshores tenim $E \leq H(h, \mu, n)$. De l'expressió anterior es dedueix que l'error generalitzat creix proporcionalment amb la dimensió VC del classificador.

Considerem ara el conjunt de separadors $F_\rho \subset F$ que conté els separadors lineals amb marge ρ. En aquest cas:

$$\text{dimVC}(F_\rho) \leq \min\left(\left\lceil \frac{R^2}{\rho^2} \right\rceil, d\right) + 1$$

Per a $R = \max_{\forall x_i \in D} \|x_i\|$. Per tant, un augment del marge provoca una disminució de la dimensió VC dels classificadors, i aconsegueix una disminució de l'error generalitzat.

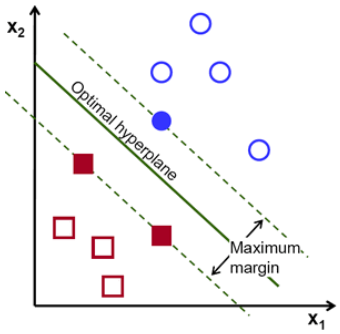

Figura 3.4: Hiperplà separador òptim [1]

Així, podem definir l'**hiperplà separador òptim** com aquell hiperplà canònic que classifica correctament les dades i té el major marge possible. Aquest serà el que té la major probabilitat de mostrar un menor nivell d'error generalitzat.

3.2.4. Regularització i violacions del marge

Tot i que a l'apartat anterior s'ha mostrat que l'hiperplà separador òptim és el que té més probabilitat de tenir un menor nivell d'error generalitzat, la maximització del marge pot no ser un criteri suficient per a obtenir un bon rendiment.

El fet és que, amb dades amb molt de soroll (o conjunts no separables linealment), el hiperplà separador òptim pot no ser una bona tria, o fins i tot pot no existir. Aquest aspecte es pot solventar mitjançant un sistema de **regularització** basat en **violacions del marge**.

Anomenem **regularització** a qualsevol mètode utilitzat per a controlar explícitament la complexitat d'un model donat. En aquest cas, la complexitat del model depèn del marge geomètric, així que per a regularitzar-lo s'ha de poder modificar aquesta magnitud.

En un hiperplà separador òptim, però, aquesta quantitat és fixa. Per aquesta raó, es permet l'aparició de certes **violacions del marge**, és a dir, certs punts que apareixen dins del marge, però que no les tenim en compte.

Així, quantes més dades es puguin ignorar, major serà el marge geomètric, i menor serà l'error de generalització. Es pot donar el cas, però, que un gran nombre de violacions del marge faci augmentar l'error d'entrenament. Això també fa disminuir l'error de generalització, de manera que cal regular adequadament aquest equilibri.

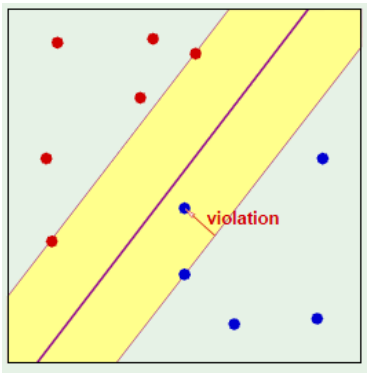

Figura 3.5: Violacions del marge [1]

Per tant, en la tria del marge hi ha un compromís entre l'**ajustament de les dades** (separació dels punts) i la **complexitat del model** (marge).

3.2.5. Entrenament de SVM

El procés d'entrenament d'una SVM consisteix en trobar un **hiperplà separador òptim** que classifiqui correctament les dades. La idea, a més, és aplicar un Feature Map $\phi: D \rightarrow H$. Un hiperplà classificarà correctament les dades si, per a totes les dades x_i compleix:

$$\begin{cases} \langle w, \phi(x_i) \rangle + w_0 > 0 & si\ t_i = +1 \\ \langle w, \phi(x_i) \rangle + w_0 < 0 & si\ t_i = -1 \end{cases}$$

Aquesta expressió la podem re-escriure en una sola equació:

$$t_i(\langle w, \phi(x_i) \rangle + w_0) > 0$$

D'entre tots els hiperplans que compleixen aquesta expressió, volem trobar aquell que minimitzi el marge ρ. Si considerem l'hiperplà canònic, podem reduir el problema a:

$$\pi_{opt}: \max_{w,w_0} \frac{1}{\|w\|}$$
$$\text{Subjecte a } 1 - t_i(\langle w, \phi(x_i) \rangle + w_0) \leq 0$$

Com que aquest problema d'optimització és molt complex i no té fàcil solució, habitualment es resol el problema següent, que és equivalent:

$$\pi_{opt}: \min_{w,w_0} \frac{1}{2} \|w\|^2$$
$$\text{Subjecte a } 1 - t_i(\langle w, \phi(x_i) \rangle + w_0) \leq 0$$

Aquest és un problema quadràtic (*Quadratic Problem* – QP) en *w*, i és més senzill de resoldre. Aquest model, però, no permet cap mena de violació del marge, i per tant s'anomena *hard margin* SVM [5].

Per a permetre l'aparició de violacions del marge, definim un conjunt de variables auxiliars ε_i que compleixen:

$$\varepsilon_i = \begin{cases} \rho - d(\phi(x_i), \pi) & si\ d(\phi(x_i), \pi) < \rho \\ 0 & altrament \end{cases}$$

Aquestes variables s'anomenen *slack variables*, i indiquen la magnitud de la violació del marge de cada dada del domini. Així, si permetem les violacions del marge, el nostre hiperplà haurà de complir per a totes les dades x_i:

$$t_i(\langle w, \phi(x_i)\rangle + w_0) \geq 1 - \varepsilon_i$$

El problema és que la magnitud d'aquestes variables ε_i no està acotada enlloc. Per a regular la magnitud d'aquestes violacions, s'afegeix al criteri d'optimització un **paràmetre de cost** $C > 0$, que penalitza la magnitud de les violacions del marge.

Així doncs, el nostre problema d'optimització resulta:

$$\pi_{opt}: \min_{w,w_0} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \varepsilon_i$$
$$\text{Subjecte a } 1 - \varepsilon_i - t_i(\langle w, \phi(x_i)\rangle + w_0) \leq 0 \text{ i } -\varepsilon_i \leq 0$$

Cal destacar que un valor elevat del paràmetre de cost tendeix a provocar un infraajust, mentre que un valor reduït del paràmetre de cost tendeix a provocar un sobreajust.

Per a resoldre aquest problema d'optimització, cal utilitzar el concepte de **multiplicadors de Lagrange**. Així doncs, definim el **lagrangià primal** com:

$$L_P(w, w_0, \varepsilon, \alpha, \mu) = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n}(C - \alpha_i - \mu_i)\varepsilon_i + \sum_{i=1}^{n}\alpha_i[1 - t_i(\langle w, \phi(x_i)\rangle + w_0)]$$

On $\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \cdots \\ \varepsilon_n \end{pmatrix}$, $\alpha = \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_n \end{pmatrix}$ i $\mu = \begin{pmatrix} \mu_1 \\ \cdots \\ \mu_n \end{pmatrix}$.

A l'expressió anterior, les variables w, w_0, ε són les anomenades **variables primals**, i corresponen a les variables objectiu del problema original. Les variables α, μ són les anomenades **variables duals**, i corresponen als multiplicadors de Lagrange que acompanyen a les restriccions imposades. Aquest lagrangià és equivalent al problema anterior:

$$\min_{w,w_0,\epsilon,\alpha,\mu} L_P(w, w_0, \varepsilon, \alpha, \mu)$$
$$\text{Subjecte a } \alpha_i \geq 0 \text{ i } \mu_i \geq 0$$

Per a obtenir la solució del Lagrangià primal, cal derivar respecte de les variables primals:

$$\frac{\partial L_P(w, w_0, \varepsilon, \alpha, \mu)}{\partial w} = w - \sum_{i=1}^{n} \alpha_i t_i \phi(x_i)$$
$$\frac{\partial L_P(w, w_0, \varepsilon, \alpha, \mu)}{\partial w_0} = -\sum_{i=1}^{n} \alpha_i t_i$$
$$\frac{\partial L_P(w, w_0, \varepsilon, \alpha, \mu)}{\partial \varepsilon_i} = C - \alpha_i - \mu_i$$

Si igualem a 0 les expressions anteriors, obtenim:

$$w = \sum_{i=1}^{n} \alpha_i t_i \phi(x_i)$$
$$\sum_{i=1}^{n} \alpha_i t_i = 0$$
$$C = \alpha_i + \mu_i$$

Re-introduir aquestes expressions dona lloc a l'anomenat **lagrangià dual**:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j t_i t_j \langle \phi(x_i), \phi(x_j) \rangle$$

Així, el problema d'optimització es redueix a minimitzar $L_D(\alpha)$, que és també un QP en α. Aquesta formulació per a la SVM s'anomena *soft margin* SVM [5]. Cal destacar que les dades del domini només apareixen en la formulació del problema en forma de producte escalar. Així doncs, podem aplicar el *kernel Trick* per a treballar amb mètodes kernel, i el lagrangià resulta de la forma:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j t_i t_j k(x_i, x_j)$$

Aquesta expressió es pot escriure en termes de matrius, de manera molt més clara:

$$\min_{\alpha} e^T \alpha - \frac{1}{2} \alpha^T H \alpha$$
Subjecte a $0 \leq \alpha \leq C$ i $\alpha^T t = 0$

On $(H)_{i,j} = t_i (K)_{i,j} t_j$ i e és un vector de dimensió n amb valor 1 a totes les components.

3.2.6. Classificació amb SVM

Com s'ha vist a l'apartat anterior, l'entrenament d'una SVM consisteix en la resolució del problema quadràtic següent:

$$\min_{\alpha} e^T\alpha - \frac{1}{2}\alpha^T K\alpha$$
$$\text{Subjecte a } 0 \leq \alpha_i \leq C \text{ i } \alpha^T t = 0$$

La solució d'aquest problema serà un vector de la forma $\alpha = \begin{pmatrix}\alpha_1 \\ \cdots \\ \alpha_n\end{pmatrix}$. Tot i que és possible trobar diverses solucions per al vector α, totes elles porten a la mateixa expansió del vector *w*. A més, un dels principals avantatges de la SVM es que aquesta solució és *sparse*, és a dir, que una quantitat significativa dels elements del vector són nuls.

Com que les dades es troben multiplicades pel seu coeficient α_i, totes aquelles dades els coeficients dels quals s'anul·len són irrellevants per a la solució del problema. Així doncs, només és necessari conservar el conjunt de dades $S = \{x_i \in D | \alpha_i > 0\}$. Aquest conjunt de dades s'anomenen **vectors suport** [5]. A més, definim $S_+ = \{x_i \in S | t_i = 1\}$ i $S_- = \{x_i \in S | t_i = -1\}$ com els conjunts de **vectors suport positius** i **vectors suports negatius**, respectivament.

La *sparsity* de la solució és deguda a l'efecte de les variables α_i. Per una banda tenim $C = \alpha_i + \mu_i$, però d'altra banda tenim que $\mu_i = 0$ si i només si $\varepsilon_i > 0$. Per tant, si $\varepsilon_i > 0$ tindrem $\alpha_i = C$. A més, $\alpha_i = 0$ si i només si $t_i(\langle w, \phi(x_i)\rangle + w_0) < 1 - \varepsilon_i$. Per tant, si una dada es troba fora del marge tindrem $\alpha_i = 0$. Així, podem realitzar la classificació següent:

Valor de α_i	Tipus de vector	És vector suport?
$\alpha_i = 0$	Fora del marge	No
$0 < \alpha_i < C$	Al marge	Si
$\alpha_i = C$	Dins del marge	Si

Taula 3.1: Classificació de vectors suport

Ara, cal destacar que la recuperació del vector *w* és impossible, ja que aquest pertany a l'espai *H* (que no es construeix mai explícitament). En canvi, podem considerar el vector α com la representació de *w* en la base de l'espai *H* formada pels elements del domini.

Així, podem adaptar el criteri de classificació per a poder treballar amb aquesta representació. La funció de classificació original és:

$$f(x) = \text{sgn}(\langle w, \phi(x)\rangle + w_0)$$

Si tenim en compte l'expansió del vector *w*, podem escriure:

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i t_i \, k(x_i, x) + w_0\right)$$

A més, només és necessari utilitzar aquells vectors que siguin vectors suport. El que sí que és necessari és computar el valor de w_0, que es pot obtenir a partir d'aquells punts amb $0 < \alpha_i < C$ (els punts que es troben al marge). Definim els conjunts $\bar{S} = \{x_i \in D | 0 < \alpha_i < C\}$, $\bar{S}_+ = \{x_i \in \bar{S} | t_i = 1\}$ i $\bar{S}_- = \{x_i \in S | t_i = -1\}$. En aquests punts, es compleix $\varepsilon_i = 0$, i per tant podem escriure:

$$w_0 = t_i - \sum_{j=1}^{n} \alpha_j t_j k(x_j, x_i)$$

Si sumem tots els termes dels diferents punts al marge, podem obtenir una expressió numèricament més estable per al terme independent [5]:

$$w_0 = \frac{|\bar{S}_+| - |\bar{S}_-|}{|\bar{S}|} - \frac{1}{|\bar{S}|} \sum_{\forall x_i \in \bar{S}} \sum_{j=1}^{n} \alpha_j t_j k(x_j, x_i)$$

3.2.7. SVM amb kernel lineal en 1D

Tot i que les SVM genèriques (en qualsevol dimensió d'entrada i amb qualsevol kernel) tenen una complexitat elevada degut al coll d'ampolla de la resolució del QP, la resolució d'una SVM amb un kernel lineal en un espai de 1D és un problema molt més senzill [23] [24].

Cal destacar que en aquest problema podem treballar amb l'expressió primal del lagrangià, ja que $w \in \mathbb{R}$. Així, tots els càlculs es realitzaran en Input Space.

Suposem que tenim un conjunt de dades $(x_1, t_1), (x_2, t_2), [\dots], (x_n, t_n) \in \mathbb{R} \times \{\pm 1\}$. En base a les propietats descrites anteriorment per a una SVM genèrica, i al fet que treballem en l'espai d'entrada d'una dimensió, es compleixen les propietats següents:

1) Si x_i és classificat correctament, i $t_i(w \cdot x_i + w_0) > 1$, aleshores $\varepsilon_i = \alpha_i = 0$.

2) Per a tot punt x_i, si $\varepsilon_i > 0$, aleshores $\mu_i = 0$, $\alpha_i = C$ i $\varepsilon_i = 1 - t_i(w \cdot x_i + w_0)$.

3) Sigui α^+ la suma de tots els valors α_i tals que $t_i = 1$, α^- la suma de tots els valors α_i tals que $t_i = -1$, n^+ el nombre de vectors suport positius amb $\varepsilon_i > 0$ i n^+ el nombre de vectors suport negatius amb $\varepsilon_i > 0$. Aleshores, es compleix l'expressió $\alpha^+ - \alpha^- + C(n^+ - n^-) = 0$.

4) En ser les dades unidimensionals, tots els vectors suport positius tenen la mateixa coordenada. De la mateixa manera, tots els vectors suport negatius tenen també la mateixa coordenada.

5) En ser les dades unidimensionals, es compleix $n^+ = n^-$ i $\alpha^+ = \alpha^-$.

6) Donat un model SVM amb un conjunt de vectors suport positius, com que tots tenen la mateixa coordenada, el model és equivalent a que un d'ells tingui $\alpha_i = \alpha^+$ i els altres $\alpha_j = 0$. El mateix es pot aplicar a la classe negativa. Per tant, es pot considerar que el sistema té un únic vector suport positiu al marge i un únic vector suport negatiu al marge.

Per aquestes raons, a priori ja podem limitar el nombre de casos: Només cal validar totes les possibles parelles entre classes (positiva i negativa). Cal tenir en compte, però, que per a cada punt *p* de la classe positiva, existeix un únic punt *q* que satisfà la propietat 5).

Així doncs, només és necessari recórrer els elements de la classe positiva, identificar les seves parelles a la classe negativa i computar el valor del seu lagrangià explícitament. Aquella parella que resulti en un valor òptim del lagrangià serà assignada com a vectors suport del marge, i s'utilitzaran per a computar els valors de *w* i w_0.

Sigui p_i el i-èssim punt de la classe positiva ordenada de menor a major, i $d_i^+ = \sum_{k=i}^{n_+} p_k$. De manera anàloga, definim q_j com el j-èssim punt de la classe negativa ordenada de major a menor, i $d_i^- = \sum_{k=i}^{n_-} q_k$.

Tal com estan definits, si p_i és un vector suport al marge, aleshores q_i és la seva parella. Això permet computar de manera explícita els següents valors:

- Com que p_i i q_i són vectors del marge, compleixen $w \cdot p_i + w_0 = 1$ i $w \cdot q_i + w_0 = -1$. Per tant, tenim $w = \frac{2}{p_i - q_i}$.

- De les propietats 2) i 5), podem deduir que $\sum_{i=1}^{n} \varepsilon_i = 2(n_+ - i) - w(d_i^+ - d_i^-)$.

- El valor del lagrangià primal es pot computar explícitament: $L = \frac{w^2}{2} + C \cdot \sum_{i=1}^{n} \varepsilon_i$. En el nostre cas, cada valor es computa $L_i = \frac{2}{(p_i - q_i)^2} + C \cdot \big(2(n_+ - i) - w(d_i^+ - d_i^-)\big)$.

El nostre objectiu, doncs, és trobar $\min_i L_i$ de manera explícita, i computar $w = \frac{2}{p_i - q_i}$ i $w_0 = -w\frac{p_i + q_i}{2}$ directament.

Si utilitzem un algoritme d'ordenació lineal (com Radix Sort), el cost temporal de la resolució de la SVM per aquest mètode és $O(n)$

3.3. Kernel FDA

3.3.1. Projecció amb KFDA

El mètode FDA (*Fisher Discriminant Analysis*) [25] és una tècnica que pretén trobar una direcció *w* a l'espai que separi el millor possible dades de dues classes identificades. Això ho aconsegueix maximitzant el criteri següent:

$$J(w) = \frac{w^T \mathrm{S}_b w}{w^T \mathrm{S}_w w}$$

On S_b i S_w són les **matrius de dispersió** (*Scattering*) **intra classe** (*between class*) **i inter classe** (*within class*), respectivament. Per tal d'aplicar aquest criteri a **Feature Space**, definim els següents vectors i matrius (estenent les definicions de FDA) [5] [26]:

$$D_+ = \{x_i \in D | t_i = +1\} \qquad n_+ = |D_+|$$
$$D_- = \{x_i \in D | t_i = -1\} \qquad n_- = |D_-|$$

$$\mu_+ = \frac{1}{n_+} \sum_{\forall x_i \in D_+} \phi(x_i)$$
$$\mu_- = \frac{1}{n_-} \sum_{\forall x_i \in D_-} \phi(x_i)$$
$$\mathrm{S}_b = (\mu_+ - \mu_-)(\mu_+ - \mu_-)^T$$

$$\mathrm{S}_+ = \sum_{\forall x_i \in D_+} (\phi(x_i) - \mu_+)(\phi(x_i) - \mu_+)^T$$
$$\mathrm{S}_- = \sum_{\forall x_i \in D_-} (\phi(x_i) - \mu_-)(\phi(x_i) - \mu_-)^T$$
$$\mathrm{S}_w = \mathrm{S}_+ + \mathrm{S}_-$$

A més, definim les matrius K_+ i K_- com les matrius de kernel de mida $n \times n_+$ i $n \times n_-$ obtingudes a partir de les funcions de kernel de les dades respecte els elements de les classes positiva i negativa, respectivament.

A més, suposarem que el vector objectiu té una expansió de la forma:

$$w = \sum_{i=1}^{n} \alpha_i \phi(x_i)$$

Ara, definim els vectors:

$$m_+ = \frac{1}{n_+} e^T K_+ \qquad m_- = \frac{1}{n_-} e^T K_-$$

Així, podem escriure $w^T \mathrm{S}_b w = \alpha^T M_J \alpha$, amb:

$$M_J = (m_+ - m_-)(m_+ - m_-)^T$$

De manera similar, podem escriure $w^T \mathrm{S}_w w = \alpha^T N_J \alpha$, amb:

$$N_J = K_+ \left(I - 1_{n_+}\right) K_+^T + K_- \left(I - 1_{n_-}\right) K_-^T$$

On 1_n és una matriu on tots els elements tenen valor $\frac{1}{n}$.

Així, podem re-escriure el criteri a optimitzar com:

$$J(\alpha) = \frac{\alpha^T M_J \alpha}{\alpha^T N_J \alpha}$$

El vector α que maximitzi aquesta expressió serà aquell que separi millor les dades. Per a optimitzar-la, derivem l'expressió anterior i igualem a 0, obtenint:

$$\frac{\partial J(\alpha)}{\partial \alpha} = \frac{\left(2M_J \alpha\right)\left(\alpha^T N_J \alpha\right) - \left(2N_J \alpha\right)\left(\alpha^T M_J \alpha\right)}{\left(\alpha^T N_J \alpha\right)^2}$$

$$\left(N_J^{-1} M_J\right)\alpha = \left(\frac{\alpha^T M_J \alpha}{\alpha^T N_J \alpha}\right)\alpha$$

És a dir, que el criteri anterior té extrems relatius en tots aquells vectors que siguin vectors propis de la matriu $N_J^{-1} M_J$. Si manipulem una mica l'expressió anterior, resulta:

$$\left(N_J^{-1} M_J\right)\alpha = J(\alpha) \cdot \alpha$$

Com que el nostre objectiu és maximitzar $J(\alpha)$, voldrem trobar el vector propi associat al major valor propi de la matriu $N_J^{-1} M_J$. Es pot demostrar que aquest vector ha de complir:

$$\alpha \propto N_J^{-1}(m_+ - m_-)$$

Així doncs, qualsevol vector que sigui proporcional a aquesta direcció realitzarà una separació probabilísticament òptima de les dades. Per simplicitat, triem:

$$\alpha = N_J^{-1}(m_+ - m_-)$$

Aquesta expressió té dos problemes:

1) Fins i tot si la matriu té rang màxim, és possible que aquesta no es pugui invertir degut a errors numèrics. És per aquesta raó que se sol aplicar una **regularització** a la matriu, de manera que treballem amb:

$$N_\varepsilon = N_J + \varepsilon \cdot I \qquad \alpha = N_\varepsilon^{-1}(m_+ - m_-)$$

Per a un cert valor ε donat. Aquest valor ha de ser prou gran com per a aconseguir evitar problemes numèrics a la inversió, però prou petit com per a no fer variar significativament el resultat de la inversió de la matriu N_ε.

2) En ser la matriu N_J simètrica i semi-definida positiva, és possible que tot i aplicar la regularització, aquesta no tingui rang màxim.

En el segon cas, és necessari treballar amb la matriu pseudo-inversa de N_J, en comptes d'utilitzar la matriu inversa:

$$\alpha = N_\varepsilon^{+}(m_+ - m_-)$$

El procés d'obtenir la matriu pseudo-inversa pot ser molt costós, però es pot accelerar degut al fet que la nostra matriu és simètrica i semi-definida positiva [27]. En no ser definida positiva, no podem aplicar la descomposició LL^T de Txoleski, ja que aquesta requereix que tots els valors propis siguin positius.

En comptes d'això, podem aplicar la descomposició LDL^T, una variant de la descomposició LL^T on la matriu D és diagonal. Aquest mètode té l'avantatge que no requereix que la matriu sigui definida positiva, sinó que n'hi ha prou que sigui simètrica (o que tingui tots els valors propis reals).

Un cop obtinguda la descomposició LDL^T hem de tenir en compte que tots els elements de la diagonal principal seran 0 o positius. Així, suposem que hem obtingut una descomposició en blocs de la forma:

$$L = \begin{pmatrix} L_{1,1} & 0 \\ L_{2,1} & L_{2,2} \end{pmatrix} \qquad D = \begin{pmatrix} D_{1,1} & 0 \\ 0 & 0 \end{pmatrix}$$

Si considerem la matriu $\bar{L} = \begin{pmatrix} L_{1,1} \\ L_{2,1} \end{pmatrix}$, podem escriure:

$$LDL^T = \bar{L} D_{1,1} \bar{L}^T$$

Ara, si considerem la matriu $\bar{D} = \bar{L}^T \bar{L} D_{1,1} \bar{L}^T \bar{L}$, aleshores podem computar:

$$(LDL^T)^+ = \bar{L} \bar{D}^{-1} \bar{L}^T$$

Com que la matriu $\bar{D}$ serà sempre simètrica i definida positiva, podem computar la matriu $\bar{D}^{-1}$ utilitzant la descomposició de Txoleski. D'aquesta manera, podem computar la matriu pseudo-inversa mitjançant dues descomposicions de Txoleski, sent la segona descomposició potencialment més ràpida que la primera (ja que la matriu $\bar{D}$ té rang igual al nombre de valors propis de la matriu diferents a 0).

Així, el procediment per a calcular la matriu N_ε^+ de manera eficient seria el següent:

1) Si la matriu N_ε té rang màxim (tots els valors propis són positius), computem la matriu inversa mitjançant la descomposició LL^T de Txoleski.

2) Si la matriu N_ε té algun valor propi nul, computem la matriu pseudo-inversa mitjançant la descomposició LDL^T de Txoleski i el procediment anterior.

3.3.2. Classificació amb KFDA

Tot i que el mètode FDA (i KFDA, per extensió) troba la direcció òptima de separació de les dades, no indica la manera òptima de separar-les explícitament. Per aquesta raó, es pot acompanyar la projecció FDA (o KFDA) amb una classificació ajustant un LDA (*Linear Discriminant Analysis*) de les dades projectades [5] [26]. Aquesta tècnica es basa en la suposició de la normalitat de les dades (projectades), i en triar l'hiperplà que tingui la major probabilitat de separar adequadament les dades.

Primer de tot, suposem que hem obtingut un conjunt de dades projectades de la forma:

$$Z = \alpha^T K \qquad Z_+ = \{z_i \in Z | t_i = +1\} \qquad Z_- = \{z_i \in Z | t_i = -1\}$$

Per tant, tenim un conjunt de *n* projeccions de la forma z_i. El mètode LDA obté per a aquests punts un separador de la forma:

$$\pi: w \cdot z + w_0 = 0$$

Per a calcular els coeficients del separador, definim:

$$\bar{z}_+ = \frac{1}{n_+} \sum_{\forall z_i \in Z_+} z_i$$

$$\bar{z}_- = \frac{1}{n_-} \sum_{\forall z_i \in Z_-} z_i$$

$$\bar{\sigma}_+ = \frac{1}{n_+ - 1} \sum_{\forall z_i \in Z_+} (z_i - \bar{z}_+)^2$$

$$\bar{\sigma}_- = \frac{1}{n_- - 1} \sum_{\forall z_i \in Z_-} (z_i - \bar{z}_-)^2$$

$$\sigma = \frac{n_+}{n}\sigma_+ + \frac{n_-}{n}\sigma_-$$

Ara, els coeficients òptims del separador es defineixen per:

$$w = \frac{\bar{z}_+ - \bar{z}_-}{\sigma}$$
$$w_0 = \frac{\bar{z}_-^2 - \bar{z}_+^2}{2\sigma} + \ln\left(\frac{n_+}{n_-}\right)$$

Per a classificar un nou punt, apliquem la funció de classificació:

$$f(x) = \text{sgn}\left(w \cdot \sum_{i=1}^{n} \alpha_i \, k(x_i, x) + w_0\right)$$

Podem simplificar la funció de classificació si re-definim:

$$\alpha = w \cdot N_\varepsilon^+(m_+ - m_-)$$
$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i k(x_i, x) + w_0\right)$$

3.3.3. Probabilitats a posteriori amb KFDA

Un dels avantatges de KFDA respecte les SVM és que es pot adaptar molt fàcilment per a oferir resultats probabilístics [5] [26]. Aquest procés consisteix primer en obtenir primer l'hiperplà classificador de l'apartat anterior:

$$\alpha = \frac{\bar{z}_+ - \bar{z}_-}{\sigma} \cdot N_\varepsilon^+(m_+ - m_-)$$
$$w_0 = \frac{\bar{z}_-^2 - \bar{z}_+^2}{2\sigma} + \ln\left(\frac{n_+}{n_-}\right)$$

Suposem que volem predir la classe d'una nova dada (x, t). Com que hem obtingut el nostre classificador mitjançant LDA, podem afirmar que:

$$\sum_{i=1}^{n} \alpha_i k(x_i, x) + w_0 = \ln\left(\frac{P(x|t=1)P(t=1)}{P(x|t=-1)P(t=-1)}\right)$$

D'aquesta manera, podem obtenir les probabilitats a posteriori utilitzant l'expressió:

$$P(t=+1|x) = \frac{1}{1 - \exp(\sum_{i=1}^{n} \alpha_i k(x_i, x) + w_0)}$$
$$P(t=-1|x) = 1 - P(t=+1)$$

3.4. Opposite Maps

De totes les tècniques que modifiquen les dades per a que una SVM estàndard pugui veure accelerat el seu rendiment, destaquem la tècnica de *opposite maps* [17] per les següents raons:

1) La modificació de les dades consisteix en un **filtratge**, és a dir, que les dades que retorna el mètode són un subconjunt de les dades d'entrada.

2) El mètode està específicament dissenyat per a treballar amb SVMs.

3) El cost de l'algoritme pot arribar a ser lineal respecte la mida de l'entrada.

Aquestes raons fan que *opposite maps* tingui uns objectius molt semblants als del nostre projecte, i per això val la pena estudiar-lo en detall.

3.4.1. Objectius del mètode

El mètode de *opposite maps* pretén realitzar un filtratge de les dades, per tal que es pugui entrenar posteriorment una SVM (o una LS-SVM) amb el resultat de filtrar aquestes dades. El principal avantatge d'aquesta idea és que no requereix en cap moment entrenar la SVM amb la totalitat (ni amb un subconjunt massa gran) de totes les dades, com realitzen altres tècniques de reducció.

La base d'aquest mètode és la selecció de certs representants de les dades en un moment determinat de l'algoritme. Aquesta selecció es pot realitzar amb qualsevol algoritme de clustering, de manera que aporta flexibilitat al mètode. De tota manera, els resultats obtinguts han mostrat ser independents de l'algoritme de clustering utilitzat, de manera que qualsevol tria per a l'algoritme de selecció serà una tria vàlida.

La idea del mètode és seleccionar un subconjunt de dades el més petit possible que contingui tots els vectors suport que utilitzi la SVM, però sense el cost d'executar-la. Amb aquesta finalitat, es pretén seleccionar aquells punts que es trobin més a prop de la classe contrària. El concepte de proximitat a la classe contrària es defineix mitjançant un càlcul realitzat a partir de les distàncies de les dades a un conjunt de representants de la classe contrària, que haurem triat a la fase de selecció.

3.4.2. Descripció del mètode

Suposem que treballem amb un conjunt de dades de la forma $(x_1, t_1), (x_2, t_2), \dots, (x_n, t_n) \in D \times \{\pm 1\}$.

L'algoritme per al mètode de *opposite maps* consisteix en els passos següents:

1) Separar les dades disponibles segons la seva classe:

$$D_+ = \{x_i \in D | t_i = +1\} \qquad D_- = \{x_i \in D | t_i = -1\}$$

2) Aplicar un algoritme de clustering a cadascun dels conjunts per separat i recuperem els prototipus.

$$C_+ = clust(D_+) \qquad C_- = clust(D_-)$$

3) Per a cada vector $x_i \in D_+$, identifiquem el seu prototipus $\mu \in C_+$ més proper. Ara, eliminem tots els elements de C_+ que no hagin estat seleccionats. Apliquem el mateix procediment per als vectors $x_i \in D_-$ i els prototipus $\mu \in C_-$. Anomenem els conjunts filtrats $\bar{C}_+$ i $\bar{C}_-$, respectivament.

4) Per a cada vector $x_i \in D_+$, identifiquem el seu prototipus $\mu \in \bar{C}_-$ més proper. Ara, eliminem tots els elements de $\bar{C}_-$ que no hagin estat seleccionats. Apliquem el mateix procediment per als vectors $x_i \in D_-$ i els prototipus $\mu \in \bar{C}_+$. Anomenem els conjunts filtrats amb la classe contraria S_+ i S_-, respectivament.

5) Per a cada prototipus $\mu_i \in S_+$, identifiquem el seu vector $x_i \in D_+$ més proper. Per a cada prototipus $\mu_i \in S_-$, identifiquem el seu vector $x_i \in D_-$ més proper. Anomenem S el conjunt resultant de la unió de tots els vectors obtinguts.

L'algoritme determina S com el conjunt de dades filtrades amb el que s'ha d'entrenar la hipotètica SVM (o LS-SVM).

3.4.3. Resultats empírics

En aquest treball analitzarem els resultats del mètode *opposite maps* per a les SVM, a fi de poder realitzar una comparació amb el nostre mètode. En tots els casos s'ha resolt el QP de la SVM amb el mètode de SMO, ja que és un dels que més es beneficia d'eliminar del domini aquells vectors que no són vectors suport.

Els resultats que es mostren són aquells que utilitzen com a algoritme de clustering **K-means** per als kernels lineals i **Kernel K-means** per als kernels no lineals. En tots els datasets, el 80% de les dades ha estat utilitzat per a l'entrenament, i el 20% per a la validació. Els paràmetres han estat triats de manera exhaustiva, i només es mostren aquells que han mostrat millor rendiment.

En total, s'utilitzen tres datasets diferents: *Vertebral Column Pathologies* – VCP, *Breast Cancer* – BC i *Pima Indian Diabetes* – PID. Els resultats mostrats s'utilitzaran per a ser comparats amb els del nostre mètode.

Dataset	Kernel	Model	C	Accuracy	% Reducció
VCP	Lineal	Full	2.5	84.9	–
VCP	Lineal	Reduced	2.5	85.6	17.5%
VCP	RBF($\sigma = 1.5$)	Full	2.5	85.3	–
VCP	RBF($\sigma = 1.5$)	Reduced	2.5	84.6	19.7%
BC	Lineal	Full	0.04	97.0	–
BC	Lineal	Reduced	0.04	96.7	20.9%
BC	RBF($\sigma = 1.5$)	Full	1	97.1	–
BC	RBF($\sigma = 1.5$)	Reduced	1	96.0	40.2%
PID	Lineal	Full	2.5	76.9	–
PID	Lineal	Reduced	2.5	76.6	8.4%
PID	RBF($\sigma = 1.5$)	Full	1	77.2	–
PID	RBF($\sigma = 1.5$)	Reduced	1	75.5	10.6%

Taula 3.2: Resultats de opposite maps

3.4.4. Anàlisi de costos

Siguin $n_+ = |D_+|$, $n_- = |D_-|$, $m_+ = |C_+|$, $m_- = |C_-|$, $m = m_+ + m_-$, $\bar{m}_+ = |\bar{C}_+|$, $\bar{m}_- = |\bar{C}_-|$, $s_+ = |S_+|$ i $s_- = |S_-|$ els costos són els següents:

1) Cost espacial i temporal $O(n)$
2) El cost depèn de l'algoritme de clustering utilitzat, però oscil·la entre $O(n \cdot m)$ i $O(n^2)$ per al cost temporal i entre $O(n)$ i $O(n^2)$ per al cost espacial.
3) Cost espacial i temporal $O(n_+ \cdot m_+ + n_- \cdot m_-) = O(n \cdot m)$.
4) Cost espacial i temporal $O(n_+ \cdot \bar{m}_- + n_- \cdot \bar{m}_+) = O(n \cdot m)$.
5) Cost espacial i temporal $O(n_+ \cdot s_+ + n_- \cdot s_-) = O(n \cdot s) = O(n \cdot m)$.

En global, el cost temporal de l'algoritme oscil·la entre $O(n \cdot m)$ i $O(n^2)$, mentre que el seu cost espacial oscil·la entre $O(n)$ i $O(n^2)$.

4. Algoritme dissenyat: Filtratge per puresa

4.1. Formalització del problema

Suposem que tenim un conjunt de dades empíriques de la forma:

$$(x_1, t_1), (x_2, t_2), [\dots], (x_n, t_n) \in D \times \{\pm 1\}$$

Aquest conjunt de dades es pot dividir en dues classes diferenciades:

$$X_+ = \{x_i \in X | t_i = +1\} \qquad n_+ = |X_+|$$
$$X_- = \{x_i \in X | t_i = -1\} \qquad n_- = |X_-|$$

L'objectiu del mètode és obtenir un conjunt de dades $S \subseteq X$ tal que l'entrenament d'una SVM amb el conjunt *S* generi un model similar a l'entrenament d'una SVM amb el conjunt *X*.

Suposem que entrenem una SVM amb un conjunt de dades *X* que resulta en un conjunt de vectors suport *SV*. Sigui $x_i \in X$ tal que $x_i \notin SV$, aleshores el model obtingut en entrenar la SVM amb el conjunt *X* serà **idèntic** a l'obtingut amb el conjunt $X - \{x_i\}$.

Si portem aquest raonament a l'extrem, podem afirmar que el model obtingut en entrenar la SVM amb el conjunt *X* serà idèntic a l'obtingut amb el conjunt *SV*. Per tant, l'objectiu primari del nostre mètode és aproximar el conjunt *SV* el millor possible.

Una SVM identifica com a vectors suport aquells vectors que compleixen:

$$d(\phi(x_i), \pi) \leq \rho$$

És a dir, aquells vectors que es troben al marge, dins el marge, o fora el marge per l'extrem contrari. Per a estimar *SV*, doncs, és necessari estimar els dos hiperplans paral·lels que conformen el marge.

El problema és que la magnitud del marge resultant de la SVM no és coneguda a priori. Per aquesta raó, el que farem és estimar primer la direcció de l'hiperplà, i després computar diferents hiperplans paral·lels a partir d'un paràmetre configurable per l'usuari.

Aquest paràmetre podria ser simplement el valor numèric del marge. Aquesta tria, però, té tres inconvenients:

1) El valor del marge no es troba acotat. El marge d'una SVM pot ser arbitràriament gran o petit. A més, la sensibilitat a les variacions de valors del marge depèn també de l'espai *H* generat, i de la distribució de les dades.

2) El valor del marge és un paràmetre poc intuïtiu per a l'usuari. No hi ha cap relació exacta entre el valor del marge i el nombre de vectors suport triats, només una relació proporcional: A un major valor del marge, més vectors suport.

3) Generar dos hiperplans paral·lels només a partir del valor del marge requereix estimar l'hiperplà òptim, no només la seva direcció. Això suposa un problema afegit, i una complexitat addicional per a l'algoritme.

Per aquestes raons, s'ha decidit definir un paràmetre específic per a aquest problema. Considerem un hiperplà de la forma:

$$\pi: \langle w, \phi(x) \rangle + w_0 = 0$$

Definirem com a **puresa positiva** de l'hiperplà el nombre de punts positius que es troben per sobre de l'hiperplà respecte el total de punts per sobre de l'hiperplà:

$$P_\pi^+ = \frac{|\{x \in X_+ | \langle w, \phi(x) \rangle + w_0 \geq 0\}|}{|\{x \in X | \langle w, \phi(x) \rangle + w_0 \geq 0\}|}$$

Definirem com a **puresa negativa** de l'hiperplà el nombre de punts negatius que es troben per sota de l'hiperplà respecte el total de punts per sota de l'hiperplà:

$$P_\pi^- = \frac{|\{x \in X_- | \langle w, \phi(x) \rangle + w_0 \leq 0\}|}{|\{x \in X | \langle w, \phi(x) \rangle + w_0 \leq 0\}|}$$

Cal remarcar que hi ha infinits hiperplans que tenen la mateixa puresa: Tots els hiperplans paral·lels situats entre dos punts diferents tindran sempre la mateixa puresa. A efectes pràctics, podem considerar que tots aquests plans són iguals.

Per tant, només ens interessa considerar els hiperplans que fa variar el coeficient de puresa, és a dir, aquells que contenen un punt x_i determinat. Suposem que hem estimat adequadament la direcció *w*. L'hiperplà paral·lel que passa per x_i compleix:

$$\langle w, \phi(x_i) \rangle + w_{0,i} = 0$$

D'aquesta expressió en podem deduir que:

$$w_{0,i} = -\langle w, \phi(x_i) \rangle$$

Per tant, l'expressió de l'hiperplà que passa per x_i i té la direcció de w té l'expressió:

$$\pi_i: \langle w, \phi(x)\rangle - \langle w, \phi(x_i)\rangle = 0$$

Ara, fixat el vector w, podem definir el **coeficient de puresa d'un punt** x_i com el coeficient de puresa de l'hiperplà que té w com a vector associat i que conté x_i:

$$P_i^+ = \frac{|\{x \in X_+ | \langle w, \phi(x)\rangle \geq \langle w, \phi(x_i)\rangle\}|}{|\{x \in X | \langle w, \phi(x)\rangle \geq \langle w, \phi(x_i)\rangle\}|}$$

$$P_i^- = \frac{|\{x \in X_- | \langle w, \phi(x)\rangle \leq \langle w, \phi(x_i)\rangle\}|}{|\{x \in X | \langle w, \phi(x)\rangle \leq \langle w, \phi(x_i)\rangle\}|}$$

Així doncs, seleccionarem dos punts tals que els coeficients de puresa siguin menors que un cert valor fixat per l'usuari. Concretament, donat un valor de puresa $0 \leq p \leq 1$ ens interessa identificar els punts següents:

$$i_{max} = \arg\min_i \langle w, \phi(x_i)\rangle$$
$$\text{Subjecte a } P_i^+ \geq p$$

$$i_{min} = \arg\max_i \langle w, \phi(x_i)\rangle$$
$$\text{Subjecte a } P_i^- \geq p$$

Així, els marges del nostre hiperplà estimat seran els dos hiperplans paral·lels $\pi_{i_{max}}$ i $\pi_{i_{min}}$. Arribats a aquest punt, tenim dues opcions:

1) Podem seleccionar tots els punts que serien vectors suport a una SVM que resultés amb els marges seleccionats. Així, podem definir el conjunt de sortida com:

$$S_+ = \left\{x \in X_+ | \langle w, \phi(x)\rangle \leq \langle w, \phi\left(x_{i_{max}}\right)\rangle\right\}$$
$$S_- = \left\{x \in X_- | \langle w, \phi(x)\rangle \geq \langle w, \phi\left(x_{i_{min}}\right)\rangle\right\}$$
$$S = S_+ \cup S_-$$

2) Podem seleccionar només aquells punts compresos entre els dos hiperplans. Així, podem definir el conjunt de sortida com:

$$S = \left\{x \in X | \langle w, \phi\left(x_{i_{min}}\right)\rangle \leq \langle w, \phi(x)\rangle \leq \langle w, \phi\left(x_{i_{max}}\right)\rangle\right\}$$

Amb aquest conjunt S obtingut, s'ha d'entrenar una SVM. Cal remarcar que aquest mètode requereix que se li especifiqui la funció de kernel, de manera que per a validar diferents configuracions de kernel caldrà re-executar el mètode. El paràmetre C de la SVM, per contra, es pot validar amb les mateixes dades filtrades, ja que el mètode és independent del valor de C.

Així, utilitzar la puresa com a especificador dels hiperplans té els següents avantatges:

1) El valor del paràmetre de puresa p es troba acotat en un rang $0 \leq p \leq 1$. Generalment, el mètode pot ser molt sensible a petites variacions del valor del paràmetre, però el nombre de vectors seleccionats sempre serà inversament proporcional a la puresa especificada.

2) El valor del paràmetre de puresa és inversament proporcional al risc de perdre vectors suport. Si el mètode elimina moltes dades, però ho fa amb un valor de puresa molt gran, és poc probable que aquelles dades fossin vectors suport. Per contra, si cal un nivell de puresa molt baix per eliminar moltes dades, vol dir que aquestes dades tenien una probabilitat més gran de ser vectors suport.

3) Per a generar els dos hiperplans no cal estimar el coeficient de l'hiperplà òptim. Els dos hiperplans del marge s'estimen directament a partir de les dades i dels coeficients de puresa, sense cap altre càlcul que afegeixi complexitat al mètode.

Un cop fixat el conjunt d'entrada *X* i estimada la direcció de l'hiperplà *w*, el conjunt de sortida *S* depèn únicament del valor de *p* triat. Així doncs, es poden realitzar diverses seleccions amb diversos valors de *p*, sense que això suposi un cost addicional per a l'algoritme.

Així doncs, ens pot convenir més fixar el nombre de dades de sortida que es volen obtenir, i realitzar diverses execucions amb diferents valors de *p* fins que obtinguem una solució de la mida desitjada. A més, pot ser interessant afegir dades addicionals a la selecció final, per tal d'obtenir un conjunt de dades més balancejat. Això es faria afegint el mínim nombre de dades necessari per a igualar les classes, triant les més properes al marge.

Per últim, tot i que el mètode està ideat com un mètode de filtratge previ a l'entrenament de la SVM, s'ha hagut d'estimar la direcció de l'hiperplà *w*, de manera que seria possible realitzar una classificació directament amb aquest hiperplà. Aquest classificador probablement no serà igual d'eficient que la SVM, però es pot obtenir de manera molt més ràpida. A més, aquesta classificació es pot donar de manera probabilística, de la mateixa manera que s'obté un resultat probabilístic al mètode de FDA.

Així, l'aplicació de l'algoritme permet generar alhora dos objectes: Un conjunt *S* de dades d'entrada per a entrenar una SVM, i un classificador probabilístic ja entrenat amb les dades d'entrada.

4.2. Plantejament de l'algoritme

Per a l'execució de l'algoritme, suposarem que el conjunt de dades ha estat pre-processat. Si el domini és numèric, es recomana normalitzar les dades i escalar-les a un interval $-1 \leq x_i \leq 1$. Tot i que la normalització i el escalat de les dades no és estrictament necessari, és altament recomanable.

Un cop obtingudes les dades $(x_1, t_1), (x_2, t_2), \dots, (x_n, t_n) \in D \times \{\pm 1\}$ pre-processades (amb $X \subset D$), l'algoritme està format per les següents etapes:

1) **Fase de preparació**: Aquesta fase consisteix en, donat un conjunt de dades $(x_1, t_1), (x_2, t_2), \dots, (x_n, t_n) \in D \times \{\pm 1\}$, obtenir un conjunt de dades projectades $(z_1, t_1), (z_2, t_2), \dots, (z_n, t_n) \in \mathbb{R} \times \{\pm 1\}$. Les dades han de ser projectades en una direcció *w* que aproximi l'orientació de l'**hiperplà separador òptim**. Aquesta fase es realitza en dues etapes:

    a. **Fase de reducció**: Per a aproximar el vector *w*, no és necessari utilitzar la totalitat de les dades disponibles. En comptes d'això, n'hi ha prou d'utilitzar-ne un subconjunt $R \subseteq X$, que es pot obtenir de diverses maneres a decisió de l'usuari.

    b. **Fase de projecció**: Un cop obtingut un subconjunt $R \subseteq X$, aproximem el vector *w* utilitzant les dades d'aquest conjunt *R*, i projectem totes les dades respecte d'aquest vector. També hi ha diverses maneres de realitzar aquesta aproximació, depenent del mètode triat per l'usuari.

    D'aquesta manera, cada reducció dona lloc a una projecció diferent. Si el mètode per a obtenir la reducció té alguna component aleatòria, podem obtenir diverses seleccions i, per tant, diverses projeccions. Així doncs, és possible determinar la preparació final com la mitjana de les projeccions obtingudes.

2) **Fase de selecció**: Un cop obtingut el conjunt de dades projectades $(z_1, t_1), (z_2, t_2), [\dots], (z_n, t_n) \in \mathbb{R} \times \{\pm 1\}$, estimem els hiperplans paral·lels en funció del coeficient de puresa p. Aquests hiperplans estan associats cadascun a un punt (i_{max} i i_{min}), per tant només és necessari identificar aquests dos punts.

3) **Fase de construcció**: Un cop obtinguts i_{max} i i_{min}, construïm el conjunt de dades filtrades i el classificador probabilístic. Hi ha també criteris diferents per a construir el conjunt de sortida, també triats per l'usuari.

Aquestes tres fases són independents entre sí: Amb una mateixa preparació es poden realitzar diverses seleccions sense haver de repetir la primera fase, i amb una mateixa selecció es poden realitzar diverses construccions sense haver de repetir cap de les fases anteriors. Això permet, per exemple, aplicar el mètode amb diversos valors de puresa amb pràcticament el mateix temps d'execució que amb un sol valor.

4.3. Fase de preparació

La fase de preparació és la que rep les dades d'entrada $(x_1, t_1), (x_2, t_2), \dots, (x_n, t_n) \in D \times \{\pm 1\}$ i retorna un conjunt de dades projectades de la forma $(z_1, t_1), (z_2, t_2), [\dots], (z_n, t_n) \in \mathbb{R} \times \{\pm 1\}$. Aquest procés es realitza mitjançant l'expressió següent:

$$z_i = \langle w, \phi(x_i) \rangle$$

En la **fase de projecció**, es realitza l'estimació del paràmetre *w*. Aquesta estimació no es realitza mitjançant totes les dades, sinó que se n'utilitza un subconjunt $R \subset X$ de mida *m* obtingut durant la **fase de reducció**. Així, el vector *w* tindrà una expressió en forma d'expansió de les dades seleccionades:

$$w = \sum_{i=1}^{n} \alpha_i \phi(x_i)$$

On es compleix $\alpha_i \neq 0$ només per a aquells punts $x_i \in R$. Així, la projecció d'una dada es pot computar com:

$$z_i = \sum_{j=1}^{n} \alpha_j k(x_j, x_i)$$

La idea, a més, és obtenir la projecció definitiva com a promig de diverses projeccions. Suposem que realitzem *N* vegades les fases de reducció i projecció. Així, obtenim *N* conjunts $R^{(1)}, [\dots], R^{(N)}$ i *N* projeccions $Z^{(1)}, [\dots], Z^{(N)}$. Definim:

$$z_i^{(r)} = \sum_{j=1}^{n} \alpha_j^{(r)} k(x_j, x_i)$$

Podem calcular la mitjana de les projeccions com:

$$z_i = \frac{1}{N} \sum_{r=1}^{N} z_i^{(r)}$$

Cal destacar que aquesta projecció es pot obtenir també de manera directa a partir de les dades originals. Concretament, es pot utilitzar l'expressió:

$$z_i = \frac{1}{N} \sum_{r=1}^{N} \sum_{j=1}^{n} \alpha_j^{(r)} k(x_j, x_i)$$

Manipulant aquesta expressió podem arribar a:

$$z_i = \sum_{j=1}^{n} \left(\frac{1}{N} \sum_{r=1}^{N} \alpha_j^{(r)} \right) k(x_j, x_i)$$

D'aquesta expressió, podem deduir que el valor de α necessari per a projectar les dades com a la mitjana de les projeccions és la mitjana dels valors de α:

$$\alpha_i = \frac{1}{N} \sum_{r=1}^{N} \alpha_j^{(r)}$$

El vector α es pot computar directament:

$$\alpha = \frac{1}{N} \sum_{r=1}^{N} \alpha^{(r)}$$

I el vector Z es pot computar de dues maneres diferents:

$$Z = \frac{1}{N} \sum_{r=1}^{N} Z^{(r)} \qquad Z = \alpha^T K$$

4.3.1. Fase de reducció

La fase de reducció és la que rep les dades d'entrada $(x_1, t_1), (x_2, t_2), [\dots], (x_n, t_n) \in \mathbb{R}^d \times \{\pm 1\}$ i retorna un subconjunt de dades $R \subset D$ de mida m. La idea d'aquesta fase és que el mètode concret de reducció quedi a elecció de l'usuari, ja que qualsevol mètode que retorni un subconjunt de les dades és un mètode vàlid de reducció.

En general, considerarem que cada element del conjunt R és un **representant** d'un conjunt d'elements de D. Així doncs, podem decidir per a cada element $x_i \in D$ el seu pes δ_i, que indica el nombre d'elements als quals representa respecte el total. El pes d'un element que no pertany a R és 0. Així, el mètode de selecció en té prou de retornar un conjunt de pesos $\delta_1, \delta_2, [\dots], \delta_n \in \mathbb{R}$, amb $0 \leq \delta_i \leq 1$ i $\sum_{i=1}^{n} \delta_i = 1$. És important realitzar la reducció de manera independent per a les dues classes.

En aquest treball, hem proposat diverses alternatives per a realitzar aquesta fase:

1) **Reducció Aleatòria**: El mètode de la reducció aleatòria consisteix en decidir de manera completament aleatòria el subconjunt $R \subset D$. Aleshores, a cada punt se li assigna el representant més proper, i el pes δ_i de cada representant serà el nombre de punts que té assignats respecte al nombre total de punts.

   El cost temporal d'aquesta opció és $O(n \cdot m)$.

2) **K-means** [1]: Aquest mètode consisteix en aplicar k-means al conjunt d'entrada, amb $k = m$. El resultat d'executar k-means retorna un conjunt de representants que no necessàriament pertanyen al conjunt d'entrada, de manera que agruparem els punts segons la classe assignada i en calcularem el **medoide**.

   Definim el medoide d'un conjunt de punts com:

$$\min_{x_i \in D} \sum_{j=1}^{n} \|x_i - x_j\|$$

   Així, un medoide sempre serà un element del conjunt d'entrada. Aleshores, el pes δ_i de cada representant serà el nombre de punts que té la seva classe.

   El cost temporal d'aquesta opció és $O(n \cdot m \cdot t)$, on t és el nombre màxim d'iteracions permeses a l'algoritme de k-means.

3) **K-medoids** [1]: Aquest mètode és molt semblant al mètode de k-means, però els representants obtinguts sempre formen part del domini. Així doncs, el càlcul dels pesos és directament el nombre de punts assignats a la classe de cada representant.

   El cost temporal d'aquesta opció és $O(n^2)$.

4) **Spectral clustering** [28]: El principal inconvenient dels mètodes anteriors és que no treballen en el Feature Space de les dades, sinó en el Input Space. Això pot donar lloc a una aproximació incorrecta de les agrupacions a realitzar. Per aquesta raó, és interessant considerar algoritmes que treballin en Feature Space.

   Spectral Clustering és un algoritme basat en mètodes kernel, que es demostra equivalent a la versió kernelitzada de k-means (kernel k-means). Aquest algoritme, doncs, té el mateix problema que k-means amb els representants, i caldrà trobar-ne els medoides per a obtenir un conjunt de representants.

   El cost temporal d'aquesta opció és $O(n^3)$, i el seu cost espacial és $O(n^2)$.

5) **Fast spectral clustering** [28]: Un dels principals problemes del mètode anterior és el seu elevat cost espacial i temporal. Una manera de reduir aquest cost és aplicar aquest mètode, que consisteix en realitzar una execució de k-means amb $k = l < m$, i a continuació, aplicar spectral clustering sobre els representants obtinguts.

   Així, s'assigna a les dades associades a cada representant (obtingut via k-means) la classe assignada via spectral clustering, i els pesos δ_i s'assignen d'acord a aquesta assignació final de classes.

   El cost temporal d'aquesta opció és $O(n \cdot l \cdot t + l^3)$, i el cost espacial és $O(l^2)$.

4.3.2. Fase de projecció

La fase de projecció és la que rep subconjunt de dades $R \subset X$ de mida m i retorna un conjunt de dades projectades de la forma $(z_1, t_1), (z_2, t_2), [\dots], (z_n, t_n) \in \mathbb{R} \times \{\pm 1\}$. El conjunt R no ve donat explícitament, sinó que es dona un conjunt de pesos $\delta_1, \delta_2, [\dots], \delta_n \in \mathbb{R}$, amb $x_i \in R$ si i només si $\delta_i > 0$.

La idea d'aquesta fase és realitzar una estimació de la direcció *w* de l'hiperplà separador òptim. Aquesta estimació es fa de manera implícita, ja que no es construeix el vector *w* sino que es projecta el conjunt de dades del domini en la seva direcció. A més, és interessant garantir que el vector utilitzat per a la projecció és unitari. Això millorarà l'estabilitat de les projeccions computades com la mitjana de diverses projeccions. Per a obtenir un vector unitari només cal dividir-lo per la seva norma, de la forma:

$$\frac{\alpha}{\sqrt{\alpha^T K \alpha}}$$

En aquest treball, hem proposat diverses alternatives per a realitzar aquesta fase:

1) **SVM** [4] [5]: Una manera d'estimar la direcció *w* de l'hiperplà separador òptim és executar una SVM amb el subconjunt resultant de les dades. Aquesta SVM s'ha d'entrenar tenint en compte que les dades estan ponderades segons la reducció realitzada anteriorment.

   Si incorporem els pesos de cada mostra a les expressions de la formulació de la SVM, arribem a la següent formulació final:

   $$\min_{\alpha} e^T \alpha - \frac{1}{2} \alpha^T H \alpha$$
   $$\text{Subjecte a } 0 \leq \alpha \leq \delta C \text{ i } \alpha^T t = 0$$

   On $(H)_{i,j} = t_i t_j (K)_{i,j}$. D'aquesta manera, amb una lleugera modificació, podem obtenir una SVM que s'ajusta a les nostres necessitats.

   El principal avantatge d'aquesta tècnica és que és la més natural: és raonable suposar que l'estimació de la direcció *w* realitzada amb una SVM s'aproximarà a la direcció original.

   El seu principal inconvenient és que requereix l'especificació d'un paràmetre de cost *C*, igual que la SVM original. La determinació d'aquest paràmetre no és trivial, i tampoc hi ha manera directa de validar-ne la tria. A més, tampoc ens serviria per a estimar el valor de cost òptim de la futura SVM, ja que aquest és molt dependent del nombre i la distribució de les dades d'entrenament.

   El cost temporal d'aquesta opció és $O(m^2)$, i el cost espacial és $O(m)$.

2) **Kernel FDA** [5] [26]: Una altra manera d'estimar *w* és l'aplicació de l'algoritme KFDA. Hi ha treballs que mostren la relació entre els dos mètodes [29], de manera que és una bona tria per a realitzar l'estimació.

   Per a incorporar els pesos de cada mostra a les expressions de la formulació de KFDA [30], definim:

$$\mu_+ = \sum_{x_i \in X_+} \frac{\delta_i}{m_+/m} \phi(x_i)$$
$$\mu_- = \sum_{x_i \in X_-} \frac{\delta_i}{m_-/m} \phi(x_i)$$

   Això modifica la resta de la formulació. Si definim el vectors δ_+ com $(\delta_+)_i = \frac{\delta_i}{m_+/m}$ per a $x_i \in X_+$ i δ_- com $(\delta_-)_i = \frac{\delta_i}{m_-/m}$ per a $x_i \in X_-$, resulta:

$$m_+ = \delta_+^T K_+ \quad m_- = \delta_-^T K_-$$

   Amb aquests nous valors es pot obtenir la solució a partir de les formules definides anteriorment.

   El principal avantatge d'aquesta tècnica és la seva similitud amb les SVM, i el fet que no requereix cap mena de paràmetre de configuració per a ser utilitzat. Això permet realitzar un gran nombre de projeccions en un menor temps.

   El seu principal inconvenient és el seu temps d'execució, major que el d'una SVM. A més, a diferència de les SVMs resoltes amb SMO, KFDA si que requereix emmagatzemar la matriu de kernel completa. Això limita en certa manera el nombre de mostres que podem utilitzar en aquesta fase.

   El cost temporal d'aquesta opció és $O(m^3)$, i el cost espacial és $O(m^2)$.

3) **Projecció per KFDA simplificat (KSDA)**: Una tercera manera d'estimar *w* és realitzar una simplificació de l'algoritme KFDA. L'algoritme KFDA proposa maximitzar la següent expressió:

$$J(\alpha) = \frac{\alpha^T M_J \alpha}{\alpha^T N_J \alpha}$$

   On la matriu N_J depèn de les matrius de covariància de les classes. L'expressió final per al valor òptim de α és:

$$\alpha = N_J^{-1}(m_+ - m_-)$$

Si suposem que la matriu N_J és la matriu identitat, podem escriure:

$$\alpha = m_+ - m_-$$

Si, a més, utilitzem l'expressió amb pesos, tenim:

$$\alpha = \delta_+^T K_+ - \delta_-^T K_-$$

El principal avantatge d'aquesta tècnica és la seva similitud amb KFDA, i el seu reduït temps d'execució, mentre que el seu principal inconvenient és la possible pèrdua de similitud amb la SVM, ja que només utilitzem una petita part de la informació que aporta les dades.

El cost temporal d'aquesta opció és $O(m^2)$, i el cost espacial és $O(m)$.

4) **Projecció per centroides**: Una última manera d'estimar *w* és realitzar una simplificació encara més gran de l'algoritme KFDA. L'algoritme KFDA proposa maximitzar la següent expressió a Feature Space:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

On la matriu S_W depèn de les matrius de covariància de les classes. Seguint el mateix mètode utilitzat a KFDA, l'expressió final per al valor òptim de *w* és:

$$w = S_W(\mu_+ - \mu_-)$$

Si suposem que la matriu S_W és la identitat, resulta:

$$w = \mu_+ - \mu_- = \sum_{i=1}^{n} \left(\frac{1}{m_+} m \cdot \delta_i \alpha_i t_i\right) \phi(x_i)$$

Per tant, la direcció de projecció α equivalent al vector anterior serà:

$$\alpha_i = m\delta_i t_i \cdot \begin{cases} \frac{1}{m_+} & si\ t_i = +1 \\ \frac{1}{m_-} & si\ t_i = -1 \end{cases}$$

El principal avantatge d'aquesta tècnica la seva simplicitat, i el seu reduït temps d'execució. El seu principal inconvenient és que la seva extrema simplicitat pot reduir-ne l'eficàcia en comparació amb les altres tècniques.

El cost temporal d'aquesta opció és $O(m)$, i el cost espacial és $O(m)$.

4.4. Fase de selecció

La fase de selecció és la que rep les dades projectades $(z_1, t_1), (z_2, t_2), [\dots], (z_n, t_n) \in \mathbb{R} \times \{\pm 1\}$ i retorna els punts (i_{max} i i_{min}) associats als hiperplans que conformen el marge.

Aquests punts els decidim resolent el problema següent:

$$i_{max} = \arg\min_i \langle w, \phi(x_i) \rangle$$
$$\text{Subjecte a } P_i^+ \geq p$$

$$i_{min} = \arg\max_i \langle w, \phi(x_i) \rangle$$
$$\text{Subjecte a } P_i^- \geq p$$

On hem definit el **coeficient de puresa d'un punt** x_i com el coeficient de puresa de l'hiperplà que té *w* com a vector associat i que conté x_i:

$$P_i^+ = \frac{|\{x \in X_+ | \langle w, \phi(x) \rangle \geq \langle w, \phi(x_i) \rangle\}|}{|\{x \in X | \langle w, \phi(x) \rangle \geq \langle w, \phi(x_i) \rangle\}|}$$
$$P_i^- = \frac{|\{x \in X_- | \langle w, \phi(x) \rangle \leq \langle w, \phi(x_i) \rangle\}|}{|\{x \in X | \langle w, \phi(x) \rangle \leq \langle w, \phi(x_i) \rangle\}|}$$

I el paràmetre $0 \leq p \leq 1$ ve donat per l'usuari. Com que les dades ja ens arriben projectades en la direcció de *w* de la forma $z_i = \langle w, \phi(x_i) \rangle$, podem re-definir els criteris:

$$P_i^+ = \frac{|\{z \in Z_+ | z \geq z_i\}|}{|\{z \in Z | z \geq z_i\}|}$$
$$P_i^- = \frac{|\{z \in Z_- | z \leq z_i\}|}{|\{z \in Z | z \leq z_i\}|}$$

I el problema es redueix a:

$$i_{max} = \arg\min_i z_i$$
$$\text{Subjecte a } P_i^+ \geq p$$

$$i_{min} = \arg\max_i x_i$$
$$\text{Subjecte a } P_i^- \geq p$$

Aquest problema es pot resoldre en temps $O(n)$: Per a la classe positiva, només cal computar els valors de tots els coeficients de puresa positiva iterativament, eliminar aquelles z_i que no compleixin $P_i^+ \geq p$ i triar el mínim del conjunt resultant. El valor de la classe negativa es pot trobar de manera anàloga al de la classe positiva.

4.5. Fase de construcció

La fase de construcció és la que recull tots els resultats dels apartats anteriors (X, Z, i_{max} i i_{min}), i retorna un subconjunt $S \subseteq X$ a utilitzar per a entrenar una SVM i un classificador degudament entrenat per a classificar les dades.

4.5.1 Construcció del conjunt de sortida

Suposant que els marges obtinguts provinguessin de l'execució d'una SVM, de cara al conjunt de sortida definirem tres tipus diferents de dades:

1) Els vectors suport que es troben entre els marges.

   Aquests són els vectors x_i que compleixen $\langle w, \phi(x_{i_{min}})\rangle \leq \langle w, \phi(x_i)\rangle \leq \langle w, \phi(x_{i_{max}})\rangle$. Com que les dades ens arriben projectades, podem definir el conjunt de **vectors suport de sortida** com:

$$S_{sv} = \{x_i \in X | z_{i_{min}} \leq z_i \leq z_{i_{max}}\}$$

2) Els vectors suport que es troben fora dels marges.

   Aquests són els vectors x_i que compleixen $\langle w, \phi(x_i)\rangle \leq \langle w, \phi(x_{i_{min}})\rangle$ si $t_i = +1$ o $\langle w, \phi(x_i)\rangle \geq \langle w, \phi(x_{i_{max}})\rangle$ si $t_i = -1$. Com que les dades ens arriben projectades, podem definir el conjunt de **vectors suport fora del marge** com:

$$S_+ = \{x_i \in X_+ | z_i \leq z_{i_{min}}\}$$
$$S_- = \{x_i \in X_- | z_i \geq z_{i_{max}}\}$$
$$S_{out} = S_+ \cup S_-$$

3) Els punts que no són vectors suport, però que ens interessa incloure per tal de balancejar el conjunt de sortida o per assolir un mínim nombre de vectors del conjunt de sortida.

   Sigui $s_+ = |\{x_i \in S | t_i = 1\}|$ i $s_- = |\{x_i \in S | t_i = -1\}|$, cal distingir dos casos:

   a) Si $s_+ > s_-$, cal afegir $s = s_+ - s_-$ dades de la classe negativa. Sigui i_{down} la s-èssima dada negativa més petita que $z_{i_{min}}$, definim:

$$S_{add} = \{x_i \in X_- | z_{i_{down}} \leq z_i \leq z_{i_{min}}\}$$

b) Si $s_+ < s_-$, cal afegir $s = s_- - s_+$ dades de la classe positiva. Sigui i_{up} la s-èssima dada positiva més gran que $z_{i_{max}}$, definim:

$$S_{add} = \left\{ x_i \in X_+ | z_{i_{max}} \leq z_i \leq z_{i_{up}} \right\}$$

Els conjunts anteriors es poden modificar lleugerament per a assolir el mínim nombre de dades demanat, tot repartint-les a parts iguals entre els dos conjunts. En qualsevol cas, l'únic que cal modificar és el valor de i_{up} i i_{down}.

El còmput d'aquests conjunts té cost $O(n)$ per a tots els cassos, i no és complex obtenir-los. Per aquest motiu, és viable retornar tots els conjunts (per separat) i que sigui l'usuari el que decideixi quina combinació d'aquests conjunts vol utilitzar.

4.5.2 Construcció del classificador

De cara a la construcció del classificador, cal tenir en compte que disposem d'una projecció $(z_1, t_1), (z_2, t_2), [\dots], (z_n, t_n) \in \mathbb{R} \times \{\pm 1\}$ obtinguda amb una direcció w generada pel vector α. A més, com que les projeccions s'han obtingut a partir d'un conjunt de representants, aquest vector α serà *sparse*. Per tant, podem construir un classificador:

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i \, k(x_i, x) + w_0 \right)$$

On cal remarcar que només és necessari computar la funció de kernel amb aquells vectors que compleixen $\alpha_i \neq 0$. Així doncs, per a obtenir un classificador funcional només és necessari especificar un terme w_0 que permeti la separació de les dades.

Com que les dades han estat projectades en Feature Space, podem aplicar un algoritme lineal a les dades unidimensionals per tal d'obtenir una separació lineal de les dades projectades. Per a millorar-ne l'eficiència, aquest classificador unidimensional es pot entrenar **només** amb les dades filtrades pel mètode. En aquest document, presentem dues propostes per a aquesta fase:

1) **Classificació amb SVM en 1D amb kernel lineal**: L'execució d'una SVM amb un kernel lineal en un espai de 1D és molt eficient [23] [24], amb un cost temporal $O(n)$. Aquest mètode, per tant, classifica les dades de manera similar a una SVM. D'altra banda, requereix d'un paràmetre de configuració C que s'ha de validar per a trobar-ne el valor òptim. A més, per a obtenir probabilitats a posteriori, cal ajustar un LDA al resultat, de manera semblant a com es fa a KFDA.

2) **Classificació amb LDA en 1D**: Aquest procés és idèntic a l'utilitzat per a obtenir models de classificació per a KFDA. És molt més ràpid de computar que l'anterior (tot i que té cost temporal $O(n)$), no requereix cap paràmetre addicional i a més té l'avantatge de generar un model probabilístic de manera molt més natural [25].

4.6. Visió general

En els apartats anteriors s'ha detallat les diferents fases del mètode proposat, així com les possibles tries que es poden realitzar i la justificació de les formules utilitzades. És important, però, tenir una visió en conjunt de tot el mètode proposat. Suposem que l'entrada és un conjunt de punts de la forma $(x_1, t_1), (x_2, t_2), \dots, (x_n, t_n) \in D \times \{\pm 1\}$. Definim, a més, els següents elements:

$$X = \{x_i \in D\} \qquad T = \{t_i \in \{\pm 1\}\}$$

Addicionalment, cal que l'usuari especifiqui els següents paràmetres:

- $k: D \times D \to \mathbb{R}$ és la **funció de kernel**, especificada per l'usuari. Aquesta s'utilitza per a computar la **matriu de kernel** $(K)_{i,j} = k(x_i, x_j)$.

- $p \in \mathbb{R}$ és el **coeficient de puresa**, i ha de complir $0 \leq p \leq 1$

- $N \in \mathbb{N}$ és el **nombre de reduccions i projeccions** diferents a computar.

- $m_+, m_- \in \mathbb{R}$ són el **nombre de mostres** de la classe positiva i negativa per a la fase de reducció, respectivament.

- $red: (D \times \{\pm 1\})^n \to \mathbb{R}^n$ és la **funció de reducció**, que utilitza un dels mètodes proposats per a generar un conjunt de pesos δ que compleix $0 \leq \delta_i \leq 1$ i $\sum_{i=1}^{n} \delta_i = 1$ a partir de les dades i del nombre de representants especificat.

- $prj: (D \times \{\pm 1\} \times \mathbb{R})^n \to \mathbb{R}^n \times \mathbb{R}^n$ és la **funció de projecció**, que utilitza un dels mètodes proposats per a generar un vector de projecció α i un conjunt de dades projectades Z. El resultat d'aquesta funció ha de satisfer $Z = K\alpha$.

- $trn: (\mathbb{R} \times \{\pm 1\})^n \to \mathbb{R} \times \mathbb{R}$ és la **funció d'entrenament**, que utilitza un dels mètodes proposats per a determinar els termes w i w_0 del classificador. El terme w s'ha de multiplicar pel coeficient α per a que el terme w_0 faci la funció adequada i el resultat es pugui interpretar de manera probabilística.

El mètode retorna uns conjunts $S_{sv}, S_{out}, S_{add} \subseteq X$ com a conjunt de dades filtrades, un vector $\alpha \in \mathbb{R}^n$ i un terme independent $w_0 \in \mathbb{R}$ com a classificador probabilístic. Aquest classificador funciona amb les expressions següents:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i \, k(x_i, x) + w_0\right)$$

$$P(t = +1|x) = \frac{1}{1 - \exp(\sum_{i=1}^{n} \alpha_i k(x_i, x) + w_0)}$$

$$P(t = -1|x) = 1 - P(t = +1)$$

El mètode proposat, en forma de pseudocodi, és el següent:

1) **Per a** $r = 1..N$
   a. $\delta^{(r)} \coloneqq red(X, t, m_+, m_-)$, amb $\delta^{(r)} \in \mathbb{R}^n$
   b. $\left(\alpha^{(r)}, Z^{(r)}\right) \coloneqq prj\left(X, t, \delta^{(r)}\right)$, amb $\alpha^{(r)}, Z^{(r)} \in \mathbb{R}^n$

2) $\alpha \coloneqq \frac{1}{N}\sum_{r=1}^{N} \alpha^{(r)}$, $Z \coloneqq \frac{1}{N}\sum_{r=1}^{N} Z^{(r)}$

3) **Ordenar** *X* i *T* segons el valor de *Z*. Definim $\tilde{x}_i \in X$ i $\tilde{t}_i \in T$ com els elements de la mostra ordenats segons el valor de z_i.

4) $n_-^{\uparrow} \coloneqq n_-$, $n_+^{\uparrow} \coloneqq n_+$, $n_-^{\downarrow} \coloneqq 0$, $n_+^{\downarrow} \coloneqq 0$. Aquestes variables indicaran, en tot moment, el nombre d'elements de cada classe per sobre o per sota del punt considerat, respectivament. Suposarem que el punt inicial és per sota del mínim, de manera que es té tots els punts de les dues classes per sobre i cap per sota.

5) **Per a** $i = 1..n$
   a. **Si** $\tilde{t}_i = -1$
      i. $n_-^{\downarrow} \coloneqq n_-^{\downarrow} + 1$
      ii. $P_i^- \coloneqq \frac{n_-^{\downarrow}}{n_-^{\downarrow} + n_+^{\downarrow}}$
      iii. $P_i^+ \coloneqq \frac{n_+^{\downarrow}}{n_-^{\uparrow} + n_+^{\uparrow}}$
      iv. $n_-^{\uparrow} \coloneqq n_-^{\uparrow} - 1$
   b. **Sino** $(\tilde{t}_i = 1)$
      i. $n_+^{\downarrow} \coloneqq n_+^{\downarrow} + 1$
      ii. $P_i^- \coloneqq \frac{n_-^{\downarrow}}{n_-^{\downarrow} + n_+^{\downarrow}}$
      iii. $P_i^+ \coloneqq \frac{n_+^{\downarrow}}{n_-^{\uparrow} + n_+^{\uparrow}}$
      iv. $n_+^{\uparrow} \coloneqq n_+^{\uparrow} - 1$

6) $i_{max} \coloneqq \min\{i | P_i^- \geq p\}$, $i_{min} \coloneqq \max\{i | P_i^+ \geq p\}$

7) $S_{sv} \coloneqq \{\tilde{x}_i \in X | i_{min} \leq i \leq i_{max}\}$

8) $S_{out} \coloneqq \{\tilde{x}_i \in X | i \leq i_{min}, \tilde{t}_i = 1\} \cup \{\tilde{x}_i \in X | i \geq i_{max}, \tilde{t}_i = -1\}$

9) $s_+ = |\{\tilde{x}_i \in S_{sv} | \tilde{t}_i = 1\}|$, $s_- = |\{\tilde{x}_i \in S_{sv} | \tilde{t}_i = -1\}|$

10) **Si** $s_+ > s_-$, $S_{add} = \{\tilde{x}_i \in X | i_{max} \leq i \leq i_{max} + i_{add}, \tilde{t}_i = -1\}$

11) **Sino** (si $s_+ < s_-$), $S_{add} = \{\tilde{x}_i \in X | i_{min} - i_{add} \leq i \leq i_{min}, \tilde{t}_i = 1\}$

12) $(w, w_0) \coloneqq trn(Z, t)$, $\alpha \coloneqq w \cdot \alpha$

4.7. Anàlisi de costos

Per tal de realitzar un anàlisi dels costos asimptòtics, és important recalcar que una part de l'algoritme és configurable, de manera que és necessari conèixer els costos asimptòtics de les diferents tries.

Aquests costos es troben descrits al llarg de les seccions anteriors. Per tal de simplificar l'anàlisi, definim:

- $T_{red}(n, m)$ i $C_{red}(n, m)$ són els costos espacials i temporals, respectivament, de l'execució de la **funció de reducció**.

- $T_{prj}(n, m)$ i $C_{prj}(n, m)$ són els costos espacials i temporals, respectivament, de l'execució de la **funció de projecció**.

- $T_{trn}(n)$ i $C_{trn}(n)$ són els costos espacials i temporals, respectivament, de l'execució de la **funció d'entrenament**.

Així, els costos temporals de cada punt de l'algoritme anterior són els següents:

1) $O\left(N \cdot \left(T_{red}(n, m) + T_{prj}(n, m)\right)\right)$
   a. $O\left(T_{red}(n, m)\right)$
   b. $O\left(T_{prj}(n, m)\right)$

2) $O(N)$

3) $O(n)$ amb un algoritme d'ordenació lineal (com **Radix Sort**).

4) $O(1)$

5) $O(n)$

   [Tots els punts del 5 al 12 tenen cost $O(n)$]

12) $O(n)$

13) $T_{trn}(n)$

El cost temporal de l'algoritme, per tant, queda dominat per l'expressió:

$$O\left(n + N \cdot \left(T_{red}(n, m) + T_{prj}(n, m)\right) + T_{trn}(n)\right)$$

El cost espacial de l'algoritme és lineal o constant en tots els passos, excepte en aquells que són configurables. Aquest, per tant, queda dominat per l'expressió:

$$O\left(n + C_{red}(n,m) + C_{prj}(n,m) + C_{trn}(n)\right)$$

Si considerem només les opcions proposades en aquest treball, podem considerar els costos espacials i temporals mínims i màxims segons les tries realitzades. En concret, el cost temporal òptim seria:

$$O(N \cdot n \cdot m)$$

Aquesta configuració utilitza **reducció aleatòria**, **projecció per centroides** i qualsevol mètode d'**entrenament**, (ja que tots tenen cost $O(n)$). Si N i m prenen valors constants, aquest cost és clarament lineal.

El cost espacial òptim seria:

$$O(n)$$

Aquesta configuració utilitza **reducció aleatòria** o **k-means**, **projecció per svm** o qualsevol de les variants **reduïdes** de **KFDA**, i qualsevol mètode d'**entrenament**.

El cost temporal en el cas pitjor seria:

$$O(N \cdot n^3)$$

Aquesta configuració utilitza **spectral clustering**, qualsevol mètode de **projecció**, i qualsevol mètode d'**entrenament**.

El cost espacial en el cas pitjor seria:

$$O(n^2)$$

Aquesta configuració utilitza **spectral clustering**, qualsevol mètode de **projecció** i qualsevol mètode d'**entrenament**.

Cal destacar que els costos en cas pitjor s'obtenen només amb la utilització de spectral clustering com a tècnica de reducció. Aquest és un dels algoritmes de clustering més costosos, tant a nivell temporal com a nivell espacial. És necessari, per tant, estudiar a fons l'impacte de l'algoritme de selecció al resultat final, ja que pot esdevenir el coll d'ampolla del nostre algoritme.

4.8. Optimitzacions per a kernel lineal

L'algoritme exposat als apartats anteriors ha estat dissenyat per a treballar en el Feature Space de les dades, mitjançant projeccions i expansions de vectors com a combinació lineal de les dades. D'aquesta manera, podem aplicar l'algoritme a SVMs amb kernels no lineals de manera natural.

Tot i això, les simplificacions de la SVM amb un kernel lineal fan que, en ocasions, sigui la millor opció a triar. Això pot passar, per exemple, quan es disposa de tantes dades que fins i tot l'execució d'una SVM amb un conjunt de dades filtrades és inviable.

Per aquesta raó, és interessant considerar també la versió en Input Space de l'algoritme dissenyat. Aquest millora significativament el seu cost temporal i espacial, millorant significativament el seu rendiment respecte a la versió kernelitzada.

Si analitzem l'estructura de l'algoritme, podem veure que només requereixen de la funció de kernel els passos de la fase de preparació, ja que les altres reben directament les dades projectades. Per aquesta raó, només cal modificar aquesta primera fase.

4.8.1 Fase de preparació lineal

La fase de preparació és la que rep les dades d'entrada $(x_1, t_1), (x_2, t_2), \dots, (x_n, t_n) \in \mathbb{R}^d \times \{\pm 1\}$ i retorna un conjunt de dades projectades de la forma $(z_1, t_1), (z_2, t_2), \dots, (z_n, t_n) \in \mathbb{R} \times \{\pm 1\}$. Aquest procés es realitza mitjançant l'expressió següent:

$$z_i = \langle w, x_i \rangle$$

A diferència de com es realitza amb kernels no lineals, en aquest cas **sí** que és possible estimar el paràmetre *w* directament. Per aquesta raó, no és necessari realitzar diverses estimacions amb subconjunts de les dades, ja que sempre serà viable obtenir el vector *w* utilitzant totes les dades disponibles.

Per a projectar totes les dades, podem utilitzar l'expressió:

$$Z = w^T X$$

On entenem *X* com el conjunt de dades, col·locades per columnes.

4.8.2 Fase de reducció lineal

Degut a que les versions lineals de l'algoritme proposat no requereixen d'una reducció de les dades, la fase de reducció desapareix en la versió lineal de l'algoritme.

4.8.3 Fase de projecció lineal

La idea d'aquesta fase és realitzar una estimació de la direcció *w* de l'hiperplà separador òptim. En aquest cas, la estimació es fa de manera explícita, ja que treballem en Input Space.

De les opcions proposades, algunes les hem descartat perquè no té sentit aplicar-les. D'altres, n'hem proposat la versió lineal, que ha d'oferir els mateixos resultats.

1) **FDA** [25]: La principal manera d'estimar *w* és l'aplicació de l'algoritme FDA. Aquest té un cost temporal molt reduït, i assoleix exactament la mateixa solució que KFDA amb un kernel lineal. Per aquesta raó, és la primera opció a utilitzar si treballem amb un kernel lineal.

2) **Projecció per centroides**: Una altra manera d'estimar *w* és realitzar una simplificació de l'algoritme FDA. L'algoritme FDA proposa maximitzar la següent expressió:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

On la matriu S_W depèn de les matrius de covariància de les classes. L'expressió final per al valor òptim de *w* és:

$$w = S_W(\mu_+ - \mu_-)$$

Si suposem que la matriu S_W és la identitat, resulta:

$$w = \mu_+ - \mu_-$$

Per tant, podem computar el vector *w* directament utilitzant només els centroides de les classes. Aquest mètode és encara més ràpid que FDA, però sempre tindrà un rendiment menor que aquest. Per aquesta raó, aquest mètode es reserva només per a casos extrems, en els que fins i tot sigui inviable aplicar FDA.

4.8.4 Fase de construcció del classificador lineal

De cara a la construcció del classificador, cal tenir en compte que no utilitzarem l'expansió del vector *w* com a combinació lineal de les dades, sino que realitzarem directament la classificació com:

$$f(x) = \text{sgn}(\langle w, x \rangle + w_0)$$

Qualsevol de les tècniques descrites anteriorment per a construir el classificador són vàlides, però cal remarcar que s'utilitza directament el vector *w* per a la classificació final.

4.9. Justificació teòrica

L'objectiu del nostre mètode és realitzar una aproximació dels marges d'un hiperplà separador òptim obtingut mitjançant una SVM. Suposem que la SVM genera un hiperplà de la forma:

$$\pi: \langle w, \phi(x) \rangle + w_0 = 0$$

On el vector *w* té una expansió de la forma:

$$w = \sum_{i=1}^{n} \alpha_i \phi(x_i)$$

4.9.1. Preparació de les dades

L'objectiu d'aquest primer pas és estimar el valor de *w*. Suposem que hem realitzat una estimació de la forma:

$$v = \sum_{j=1}^{m} \beta_j \phi(c_j)$$

On c_j és el representant del clúster *j* obtingut a la fase de reducció. Com que el que volem és obtenir dos vectors idèntics (o pràcticament idèntics), minimitzem la seva distància a Feature Space:

$$\|w - v\|^2 = w^2 + v^2 - 2\langle v, w \rangle = \alpha^T K \alpha + \beta^T K_c \beta - 2\alpha^T K_{x,c} \beta$$

On K_c és la matriu de kernel $m \times m$ del conjunt de representants c_j, i $K_{x,c}$ és la matriu de kernel $n \times m$ de les dades x_i respecte els representants c_j.

Si derivem aquesta distància respecte del vector que volem aproximar i igualem a 0:

$$\frac{\partial \|w - v\|^2}{\partial \beta} = 2K_c \beta - 2K_{c,x} \alpha$$
$$K_c \beta = K_{c,x} \alpha$$

Així, l'expansió òptima del vector que aproxima *w* ha de tenir valor [5]:

$$\beta = K_c^{-1} K_{c,x} \alpha$$

Com que la matriu K_c pot no tenir rang màxim, podem treballar amb la pseudo-inversa:

$$\beta = K_c^{+} K_{c,x} \alpha$$

Suposem que hem obtingut el vector β òptim. En aquest cas, es compleix:

$$\|w - v\|^2 = \alpha^T\left(K - K_{x,c}K_c^+K_{c,x}\right)\alpha$$

Per tant, el subconjunt de les dades que minimitzi l'expressió anterior permetrà trobar la millor estimació de *w*. Cal remarcar que, si $K_{x,c}K_c^+K_{c,x} \approx K$, aleshores $\|w - v\|^2 \approx 0$.

L'objectiu de la fase de reducció, per tant, és obtenir un subconjunt de les dades tal que $K_{x,c}K_c^+K_{c,x} \approx K$. La esperança del nostre mètode és que aquesta expressió s'assoleixi mitjançant l'ús d'un algoritme de **clustering**, ja que en certa manera aquests algoritmes minimitzen la dissimilitud dels representants amb les seves dades assignades. Així, aquests haurien de donar millors resultats que la simple **selecció aleatòria**, ja que aquesta no té en compte cap mena de relació entre les dades.

Addicionalment, podem avaluar la qualitat de la selecció mitjançant l'expressió:

$$\eta_{red} = \|K - \overline{K}\|$$

On $\overline{K} = K_{x,c}K_c^+K_{c,x}$ i $\|\cdot\|$ és qualsevol norma aplicable a matrius. La qualitat de la selecció és major quant menor sigui aquest coeficient.

De cara a la projecció de les dades, en aquest cas no és tant important la exacta igualtat dels vectors *v* i *w*, ja que una simple diferència en la magnitud del vector no fa variar les posicions relatives de les dades projectades. Per aquesta raó, n'hi ha prou que els dos vectors tinguin un angle prou petit:

$$\cos\angle(w, v) = \frac{\langle w, v\rangle}{\|w\|\|v\|} = \frac{\alpha^T K_{x,c}\beta}{\sqrt{\alpha^T K\alpha}\sqrt{\beta^T K_c\beta}}$$

L'objectiu de la fase de projecció, per tant, és obtenir una aproximació del vector *w* que compleixi $\frac{\alpha^T K_{x,c}\beta}{\sqrt{\alpha^T K\alpha}\sqrt{\beta^T K_c\beta}} \approx 1$. La esperança del nostre mètode és que aquesta expressió s'assoleixi mitjançant l'ús d'una SVM entrenada amb el conjunt reduït de dades, o amb una altra tècnica que n'aproximi raonablement el resultat (com KFDA).

Addicionalment, podem avaluar la qualitat de la direcció de la projecció mitjançant l'expressió:

$$\eta_{prj} = \left|\left(\alpha^T K_{x,c}\beta\right)^2 - (\alpha^T K\alpha)(\beta^T K_c\beta)\right|$$

Altre cop, la qualitat de la projecció és major quant menor sigui aquest coeficient.

Aquests dos criteris de qualitat són purament teòrics, ja que computar-los requereix el càlcul de la matriu de kernel completa, que no sempre serà possible realitzar.

4.9.2. Selecció dels marges

Suposem ara que tenim una projecció de dades de la forma:

$$z_i = \sum_{j=1}^{n} \alpha_j k(x_i, x_j)$$

Aquesta projecció la podem obtenir en bloc si apliquem:

$$Z = \alpha^T K$$

De cara als càlculs dels coeficients de puresa, només és rellevant la posició relativa de les dades. Per aquesta raó, podem centrar i escalar les dades a conveniència. Suposem que aquestes dades projectades es divideixen en dues classes:

$$Z_+ = \{z_i \in Z | t_i = 1\} \qquad Z_- = \{z_i \in Z | t_i = -1\}$$

Aquestes dades projectades i escalades serveixen per a computar els coeficients de puresa. Aquests es calculen amb les expressions següents:

$$P_i^+ = \frac{|\{z \in Z | z \geq z_i, z \in Z_+\}|}{|\{z \in Z | z \geq z_i\}|}$$
$$P_i^- = \frac{|\{z \in Z | z \leq z_i, z \in Z_-\}|}{|\{z \in Z | z \leq z_i\}|}$$

Si realitzem una mica de manipulació dels conjunts:

$$P_i^+ = \frac{|\{z \in Z | z \geq z_i, z \in Z_+\}|}{|\{z \in Z | z \geq z_i, z \in Z_+\}| + |\{z \in Z | z \leq z_i, z \in Z_-\}|}$$
$$P_i^- = \frac{|\{z \in Z | z \leq z_i, z \in Z_-\}|}{|\{z \in Z | z \leq z_i, z \in Z_-\}| + |\{z \in Z | z \leq z_i, z \in Z_-\}|}$$

Suposant que les dades segueixen la distribució anterior, podem obtenir el valor esperat de cada coeficient de puresa en un punt donat:

$$\mathrm{E}(P_i^+) = \frac{n \cdot p(z \geq z_i, z \in Z_+)}{n \cdot p(z \geq z_i, z \in Z_+) + n \cdot p(z \geq z_i, z \in Z_-)}$$
$$\mathrm{E}(P_i^-) = \frac{n \cdot p(z \leq z_i, z \in Z_-)}{n \cdot p(z \leq z_i, z \in Z_+) + n \cdot p(z \leq z_i, z \in Z_-)}$$

Simplificant, obtenim:

$$\mathrm{E}(P_i^+) = \frac{p(z \geq z_i, z \in Z_+)}{p(z \geq z_i, z \in Z_+) + p(z \geq z_i, z \in Z_-)}$$
$$\mathrm{E}(P_i^-) = \frac{p(z \leq z_i, z \in Z_-)}{p(z \leq z_i, z \in Z_+) + p(z \leq z_i, z \in Z_-)}$$

Els coeficients de puresa que ens interessa computar, però, no són els de qualsevol punt, sinó només aquells en que es compleixi $P_i^+ \geq p$ a i_{max} o $P_i^- \geq p$ a i_{min}.

Com que el valor esperat dels coeficients de puresa conformen una funció contínua, podem suposar que en els punts i_{max} i i_{min} es complirà:

$$P_{i_{max}}^+ \approx p \qquad P_{i_{min}}^- \approx p$$

Així doncs, podem assumir:

$$\mathrm{E}\left(P_{i_{max}}^+\right) = p \qquad \mathrm{E}\left(P_{i_{min}}^-\right) = p$$

Aplicant la igualtat anterior, podem obtenir:

$$\mathrm{E}\left(P_{i_{max}}^+\right) = \frac{p\left(z \geq z_{i_{max}}, z \in Z_+\right)}{p\left(z \geq z_{i_{max}}, z \in Z_+\right) + p\left(z \geq z_{i_{max}}, z \in Z_-\right)} = p$$
$$\mathrm{E}\left(P_{i_{min}}^-\right) = \frac{p\left(z \leq z_{i_{min}}, z \in Z_-\right)}{p\left(z \leq z_{i_{min}}, z \in Z_+\right) + p\left(z \leq z_{i_{min}}, z \in Z_-\right)} = p$$

Amb una mica de manipulació, les expressions anteriors resulten:

$$\frac{p\left(z \geq z_{i_{max}}, z \in Z_+\right)}{p\left(z \geq z_{i_{max}}, z \in Z_-\right)} = \frac{p}{1-p}$$
$$\frac{p\left(z \leq z_{i_{min}}, z \in Z_-\right)}{p\left(z \leq z_{i_{min}}, z \in Z_+\right)} = \frac{p}{1-p}$$

Les expressions anteriors es poden escriure en termes de probabilitats condicionades:

$$\frac{p\left(z \in Z_+ | z \geq z_{i_{max}}\right) \cdot p\left(z \geq z_{i_{max}}\right)}{p\left(z \in Z_- | z \geq z_{i_{max}}\right) \cdot p\left(z \geq z_{i_{max}}\right)} = \frac{p\left(z \in Z_+ | z \geq z_{i_{max}}\right)}{p\left(z \in Z_- | z \geq z_{i_{max}}\right)} = \frac{p}{1-p}$$
$$\frac{p\left(z \in Z_- | z \leq z_{i_{min}}\right) \cdot p\left(z \geq z_{i_{max}}\right)}{p\left(z \in Z_+ | z \leq z_{i_{min}}\right) \cdot p\left(z \geq z_{i_{max}}\right)} = \frac{p\left(z \in Z_- | z \leq z_{i_{min}}\right)}{p\left(z \in Z_+ | z \leq z_{i_{min}}\right)} = \frac{p}{1-p}$$

Així, l'expressió $\frac{p}{1-p}$ té dues interpretacions:

- Respecte a les dades sobre el marge superior, correspon a les **odds** que una sigui positiva respecte a que sigui negativa.

- Respecte a les dades sota el marge inferior, correspon a les **odds** que una sigui negativa respecte a que sigui positiva.

D'aquesta manera, incrementar el valor de *p* té com a efecte augmentar la proporció de dades positives sobre el marge superior i la de dades negatives sota el marge inferior.

També podem escriure les probabilitats condicionades de la forma següent:

$$\frac{p\left(z \geq z_{i_{max}} | z \in Z_+\right) \cdot p\left(z_j \in Z_+\right)}{p\left(z \geq z_{i_{max}} | z \in Z_-\right) \cdot p\left(z_j \in Z_-\right)} = \frac{p}{1-p}$$
$$\frac{p\left(z \leq z_{i_{min}} | z \in Z_-\right) \cdot p(z \in Z_-)}{p\left(z \leq z_{i_{min}} | z \in Z_+\right) \cdot p(z \in Z_+)} = \frac{p}{1-p}$$

Suposem ara que les classes de les projeccions classes segueixen una distribució normal:

$$Z_+ \sim N(\mu_+, \sigma_+^2) \qquad Z_- \sim N(\mu_-, \sigma_-^2)$$

Si suposem, a més, que tenen una variància similar, podem aproximar:

$$Z_+ \sim N(\mu_+, \sigma^2) \qquad Z_- \sim N(\mu_-, \sigma^2)$$

Així, podem realitzar un escalat de la forma:

$$Z' = \frac{Z - \frac{\mu_+ + \mu_-}{2}}{\sigma^2}$$

D'aquesta manera, les dades segueixen una distribució:

$$Z_+ \sim N(\mu_+, 1) \qquad Z_- \sim N(\mu_-, 1)$$

Amb $\mu_+ + \mu_- = 0$. A més, podem assumir $\mu_- < \mu_+$. Cal remarcar, doncs, que les projeccions tenen les funcions de probabilitat següents:

$$p(z|z \in Z_+) = \frac{1}{\sqrt{2\pi}} \exp(-(z-\mu_+)^2)$$
$$p(z|z \in Z_-) = \frac{1}{\sqrt{2\pi}} \exp(-(z-\mu_-)^2)$$
$$p(z \in Z_+) = \frac{n_+}{n} \qquad p(z \in Z_-) = \frac{n_-}{n}$$

Les probabilitats acumulades de les expressions anteriors compleixen:

$$p(z \leq x | z \in Z_+) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x - \mu_+}{\sqrt{2}}\right)\right]$$
$$p(z \leq x | z \in Z_-) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x - \mu_-}{\sqrt{2}}\right)\right]$$

Si suposem $z_{i_{min}} < \mu_- < \mu_+$ podem aproximar la probabilitat acumulada [31]:

$$p(z \leq x | z \in Z_+) \approx A \cdot \exp(-B(x - \mu_+)^2)$$
$$p(z \leq x | z \in Z_-) \approx A \cdot \exp(-B(x - \mu_-)^2)$$

Per a certes constants $A, B > 0$. Si apliquem aquesta aproximació a l'expressió anterior:

$$\frac{A \cdot \exp\left(-B\left(z_{i_{min}} - \mu_-\right)^2\right) \cdot \frac{n_-}{n}}{A \cdot \exp\left(-B\left(z_{i_{min}} - \mu_+\right)^2\right) \cdot \frac{n_+}{n}} = \frac{p}{1-p}$$

Si simplifiquem l'expressió:

$$\frac{\exp\left(-B\left(z_{i_{min}} - \mu_-\right)^2\right) \cdot n_-}{\exp\left(-B\left(z_{i_{min}} - \mu_+\right)^2\right) \cdot n_+} = \frac{p}{1-p}$$

Si prenem logaritmes a ambdós costats, l'expressió resulta:

$$B\left(\left(z_{i_{min}} - \mu_+\right)^2 - \left(z_{i_{min}} - \mu_-\right)^2\right) + \ln\frac{n_-}{n_+} = \ln\frac{p}{1-p}$$

Expandint els quadrats i simplificant, resulta:

$$B\left(\mu_+^2 - \mu_-^2 - 2z_{i_{min}}(\mu_+ - \mu_-)\right) + \ln\frac{n_-}{n_+} = \ln\frac{p}{1-p}$$

Si aïllem $z_{i_{min}}$ de l'expressió anterior, resulta:

$$z_{i_{min}} = \frac{\mu_+ + \mu_-}{2} - \frac{\ln\frac{p}{1-p} - \ln\frac{n_-}{n_+}}{2B(\mu_+ - \mu_-)}$$

Si definim $L_n = \ln\frac{n_-}{n_+}$ i $L_p = \ln\frac{p}{1-p}$, i tenint en compte que $\mu_+ + \mu_- = 0$, podem simplificar l'expressió anterior:

$$z_{i_{min}} = -\frac{L_p - L_n}{2B(\mu_+ - \mu_-)}$$

D'aquesta expressió, cal destacar que com que $B > 0$ i $\mu_+ > \mu_-$, el terme $2B(\mu_+ - \mu_-)$ sempre serà positiu, i independent del valor de p. A més, el terme L_n correspon a les log-odds a priori de la classe de les dades, independents també del valor de p. Així, podem concloure:

$$z_{i_{min}} \propto \ln \frac{1-p}{p}$$

Així doncs, el marge inferior del nostre classificador tindrà una coordenada amb una magnitud inversament proporcional al valor de p, ja que la funció $f(p) = \ln \frac{1-p}{p}$ és monòtona decreixent en l'interval $0 < p < 1$.

Podem aplicar un raonament similar per a obtenir una conclusió per al marge superior, només invertint el sentit de la projecció i repetint el procediment anterior, amb un canvi de signe al final. Així, trobem:

$$z_{i_{max}} \propto \ln \frac{p}{1-p}$$

Així doncs, el marge superior del nostre classificador tindrà una coordenada amb una magnitud proporcional al valor de p, ja que la funció $f(p) = \ln \frac{p}{1-p}$ és monòtona creixent en l'interval $0 < p < 1$.

Ara, intentarem estimar el nombre de vectors resultants del filtratge. Sigui s el nombre de vectors suport, tenim:

$$s = \left|\{z_i \in Z_+ | z_i \leq z_{i_{max}}\}\right| + \left|\{z_i \in Z_- | z_i \geq z_{i_{min}}\}\right|$$

Aplicant les probabilitats obtingudes, tenim:

$$\mathrm{E}(s) = n_+ \cdot p\left(z_i \leq z_{i_{max}} | z_i \in Z_+\right) + n_- \cdot p\left(z_i \geq z_{i_{min}} | z_i \in Z_-\right)$$

Cal remarcar que $p\left(z_i \leq z_{i_{max}} | z_i \in Z_+\right)$ és més gran quant major és $z_{i_{max}}$, i que $p\left(z_i \geq z_{i_{min}} | z_i \in Z_-\right)$ és més gran quant menor és $z_{i_{min}}$. En ser el primer proporcional a p, i el segon inversament proporcional a p, podem afirmar que el valor esperat del nombre de vectors de sortida creixerà també de manera proporcional a p.

Aquesta justificació ens aporta tres idees clau:

- El paràmetre p té un efecte directe en les probabilitats condicionals dels vectors eliminats: Obliga el marge a disminuir per a garantir que es respecten les proporcions de cada classe fora els marges.

- L'amplada del marge és directament proporcional al valor del paràmetre p. Si definim el marge com $\rho = z_{i_{max}} - z_{i_{min}}$, aleshores $\rho \propto \ln \frac{p}{1-p}$, que ja hem vist que és una expressió monòtonament creixent en p.

- El paràmetre p té també una implicació directa en el nombre de vectors filtrats: com més gran és el coeficient de puresa, major serà el marge obtingut, i és més probable que nous vectors caiguin a dins.

Per aquestes raons, es justifica la tria del coeficient de puresa com a paràmetre de regulació de l'algoritme, ja que té una interpretació directa tant en termes de probabilitat com de geometria del marge i en el nombre de vectors resultants del mètode.

4.9.3. Construcció del classificador

L'objectiu d'aquesta fase és la construcció d'un classificador a partir del vector *w* de la projecció. Un dels principals problemes de construir el classificador a partir d'una direcció de projecció obtinguda només utilitzant un subconjunt de les dades és que es corre el risc que aquesta direcció pateixi un sobreajust respecte al subconjunt de dades obtingut.

Aquest problema es mitiga en gran part mitjançant el càlcul de la projecció com a mitjana de projeccions. D'aquesta manera, s'evita el problema del sobreajust i alhora es redueix el cost computacional de la projecció, ja que en molts casos és més ràpid realitzar N execucions d'un mètode amb cost $O(m)$ que executar una sola vegada el mètode amb $N \cdot m$ dades.

Cadascuna de les projeccions realitzades determina un a expansió $\alpha^{(r)}$ del vector $w^{(r)}$ obtingut. Aquesta solució és *sparse*, ja que fins i tot si no ho és en termes de les dades reduïdes, sempre ho serà en termes de dades totals (excepte si es realitza la projecció utilitzant la totalitat de les dades disponibles).

Una projecció amb *m* dades tindrà, en el cas pitjor, *m* components $\alpha_i \neq 0$. La idea, però, és que no totes les dades representen igual de bé a les dades del seu voltant, de manera que certs representants tindran una major probabilitat de ser escollits. Aquests repetirien com a representants en les diverses projeccions, fent que el nombre final de components $\alpha_i \neq 0$ sigui significativament menor que $N \cdot m$.

Un cop determinat el paràmetre α, cal determinar w_0. Aquest es determina mitjançant un entrenament en 1D que simuli la tria que podria realitzar una SVM. Per aquesta raó, no té sentit triar termes w_0 que impliquin una frontera de decisió fora dels marges indicats. Per aquesta raó, és important computar el terme w_0 utilitzant **només** les dades resultants del filtratge. Això, a més, redueix encara més el cost de l'entrenament en 1D.

4.10. Avantatges i inconvenients

Un dels principals avantatges del mètode proposat és que ha estat **específicament dissenyat** des del principi per a **treballar amb SVMs**, tot realitzant un **filtratge** de les dades. Això suposa un avantatge respecte d'altres tècniques més genèriques (com RGZ), ja que utilitzem certes justificacions teòriques basades en la pròpia SVM. A més, el mètode utilitza una **aproximació** de la SVM, mitjançant hiperplans i marges geomètrics, a diferència de *Opposite Maps*, que tot i estar dissenyada específicament per a SVMs no utilitza cap mena de similitud o aproximació a les mateixes.

Un altre dels avantatges del mètode és el **control** que es té de totes les etapes de l'algoritme: Totes elles són configurables mitjançant un o més paràmetres. A més, com que algunes d'elles tenen un cert component aleatori o el resultat depèn del valor del paràmetre, és possible realitzar diverses execucions de l'algoritme tot reciclant aquelles fases executades amb anterioritat.

Cal remarcar que l'algoritme disposa de diversos tipus de **paràmetres**: Hi ha paràmetres que regulen explícitament la **complexitat** espacial i temporal de l'algoritme (com el nombre d'elements a la fase de reducció, i hi ha paràmetres que permeten controlar directa o indirectament el **nombre de dades de sortida** (com el coeficient de puresa). Un avantatge de la tria de paràmetres realitzada és que tots tenen una interpretació directa, que permet que l'usuari realitzi (si vol) una estimació dels valors a introduir. Això no succeeix amb altres paràmetres descartats per a l'algoritme (com el marge explícit).

Cal remarcar que la tria de diverses configuracions i el control explícit de la complexitat permet que l'algoritme pugui arribar a assolir un cost **lineal** respecte la mida de l'entrada.

Una altra de les conseqüències del control explícit de la complexitat és que es pot aconseguir molt fàcilment que, per a un nombre prou gran de dades, s'aconsegueixin reduccions de temps significatives respecte l'execució d'una SVM amb la totalitat de les dades. Fins i tot si hi afegim l'entrenament d'una SVM amb les dades reduïdes, el seu temps d'execució continuarà sent significativament menor.

Com a inconvenients, cal destacar que ens podem trobar en **casos extrems** en els que el nostre algoritme no és útil: es pot donar el cas que un altre algoritme més senzill aconsegueixi la mateixa reducció amb un cost temporal molt menor, o que la selecció realitzada no permeti entrenar una SVM que classifiqui prou bé les dades.

Tot i que s'ha justificat la presa de decisions realitzada, no hi ha **cap garantia** que l'algoritme resultarà en una selecció que compleixi adequadament amb la finalitat del mètode. Tot i aquesta manca de garanties, el **bon fonament teòric** del mètode fa esperar que el seu rendiment sigui, generalment, satisfactori.

Un altre inconvenient del mètode és que, si bé el filtratge és independent del paràmetre de cost, no ho és dels paràmetres que modifiquen el kernel. Així, per a cada paràmetre de kernel que es vulgui validar, s'haurà de realitzar un filtratge independent.

5. Resultats experimentals

5.1. Descripció de les dades

Abans d'iniciar l'exposició del conjunt d'experiments realitzats, es realitzarà una breu descripció dels conjunts de dades utilitzats en aquest procés d'experimentació. És important distingir entre dos grans blocs de dades: El conjunt de **dades sintètiques**, generades localment seguint alguna certa distribució de probabilitat, i el conjunt de **dades reals**, obtingudes de diversos datasets del UCI [32] i de Kaggle [33].

5.1.1 Descripció de les dades sintètiques

Totes les dades sintètiques han estat generades localment seguint diverses distribucions de probabilitat, amb el nombre d'elements de cada classe configurable per l'usuari. Tots els conjunts són de dades bidimensionals, per tal de poder-ne realitzar una representació gràfica adequada i apreciar-ne els efectes de l'algoritme.

El conjunt més senzill de tots és l'anomenat **dataset lineal**. Aquest dataset es genera mitjançant dues distribucions normals (una per a cada classe) de la forma:

$$X_+ \sim N(\mu_+, \Sigma) \qquad X_- \sim N(\mu_-, \Sigma)$$
$$\mu_+ = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \mu_- = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

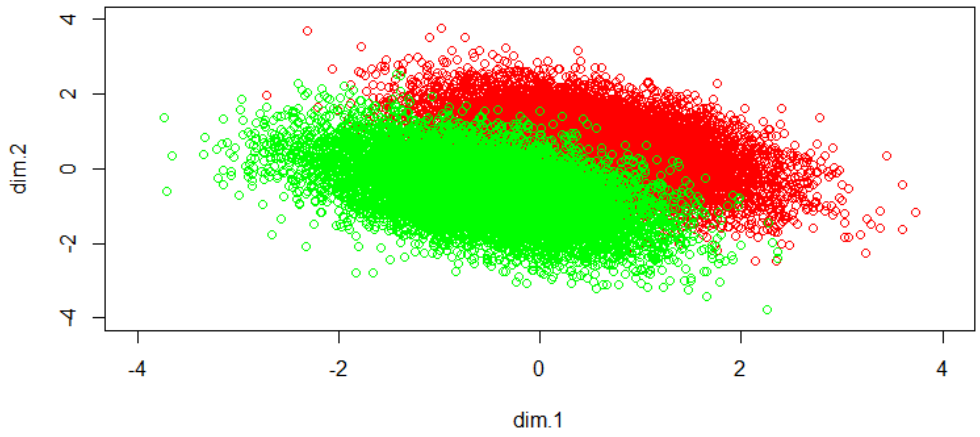

Figura 5.1: Dataset lineal amb 1000 dades a cada classe

Una variant del dataset anterior és l'anomenat **dataset quasi separable**. Aquest té una estructura molt similar al dataset lineal, però la intersecció entre les classes és molt menor. Aquest dataset es genera mitjançant dues distribucions normals de la forma:

$$X_+ \sim N(\mu_+, \Sigma) \qquad X_- \sim N(\mu_-, \Sigma)$$
$$\mu_+ = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \qquad \mu_- = \begin{pmatrix} 0 \\ -3 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

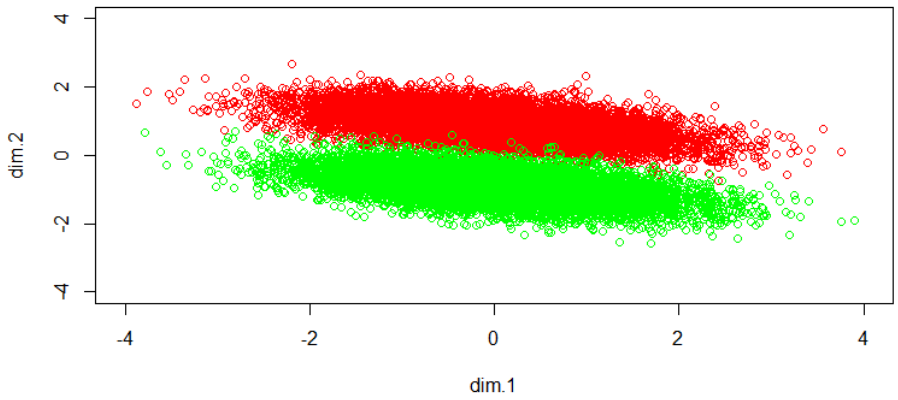

Figura 5.2: Dataset quasi separable amb 1000 dades a cada classe

Una segona variant del dataset anterior és l'anomenat **dataset separable**. Aquest té una estructura molt similar als datasets anteriors, però la intersecció entre les classes és pràcticament nul·la. Aquest dataset es genera mitjançant dues distribucions normals:

$$X_+ \sim N(\mu_+, \Sigma) \qquad X_- \sim N(\mu_-, \Sigma)$$
$$\mu_+ = \begin{pmatrix} 0 \\ 5 \end{pmatrix} \qquad \mu_- = \begin{pmatrix} 0 \\ -5 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

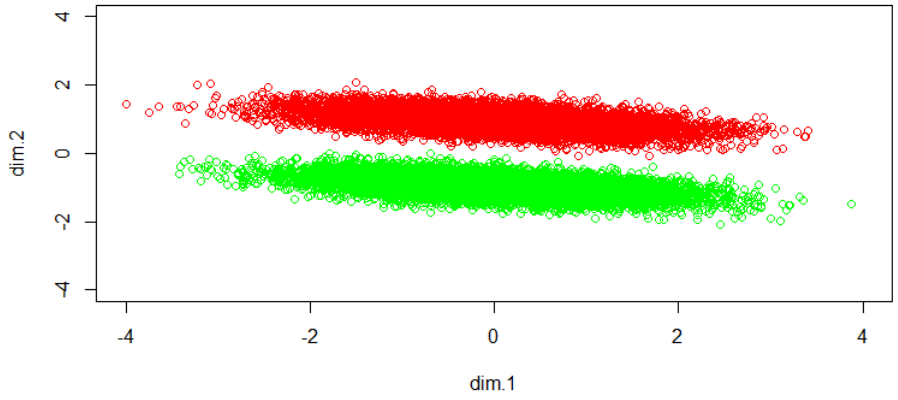

Figura 5.3: Dataset separable amb 1000 dades a cada classe

Un conjunt amb un nivell una mica més elevat de complexitat és el **dataset parabòlic**. Aquest dataset es genera mitjançant una barreja de gaussianes, seguint la distribució següent:

$$X_+ \sim \sum_{i=1}^{3} \pi_i N(\mu_i, \Sigma_i) \qquad X_- \sim N(\mu_-, \Sigma_-)$$
$$\pi_1 = 1/3 \qquad \mu_1 = \begin{pmatrix} -3 \\ 2 \end{pmatrix} \qquad \Sigma_1 = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 1 \end{pmatrix}$$
$$\pi_2 = 1/3 \qquad \mu_2 = \begin{pmatrix} 0 \\ 3 \end{pmatrix} \qquad \Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1/3 \end{pmatrix}$$

$$\pi_3 = 1/3 \qquad \mu_3 = \begin{pmatrix} 3 \\ 2 \end{pmatrix} \qquad \Sigma_3 = \begin{pmatrix} 1 & -1/2 \\ -1/2 & 1 \end{pmatrix}$$
$$\mu_- = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \Sigma_- = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

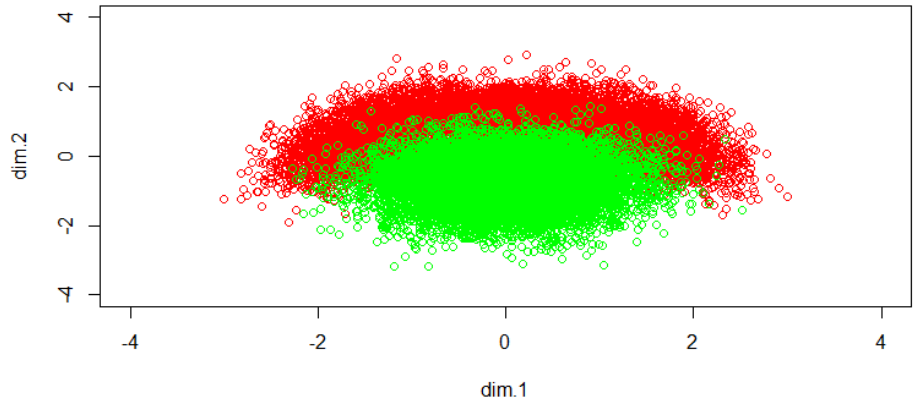

Figura 5.4: Dataset parabòlic amb 1000 dades a cada classe

Per últim, tenim el **dataset complex**. Aquest es genera d'una manera més específica:

$$X_0 \sim N(0, 2.5)$$
$$Y_+ \sim \cos(X_0) + 0.5 + N(0, 0.5) \qquad Y_- \sim \cos(X_0) - 0.5 + N(0, 0.5)$$
$$X_+ \sim \begin{pmatrix} X_0 \\ Y_+ \end{pmatrix} \qquad X_- \sim \begin{pmatrix} X_0 \\ Y_- \end{pmatrix}$$

La principal particularitat d'aquest dataset és que té una estructura molt poc lineal: Per aquesta raó, per a tractar-lo serà necessari l'ús de kernels més complexes (com RBF).

Per acabar, és important remarcar que totes les imatges han estat generades seguint la descripció aquí indicada, però han estat centrades i escalades (tal i com és recomanable que es tractin abans d'utilitzar-les en una SVM).

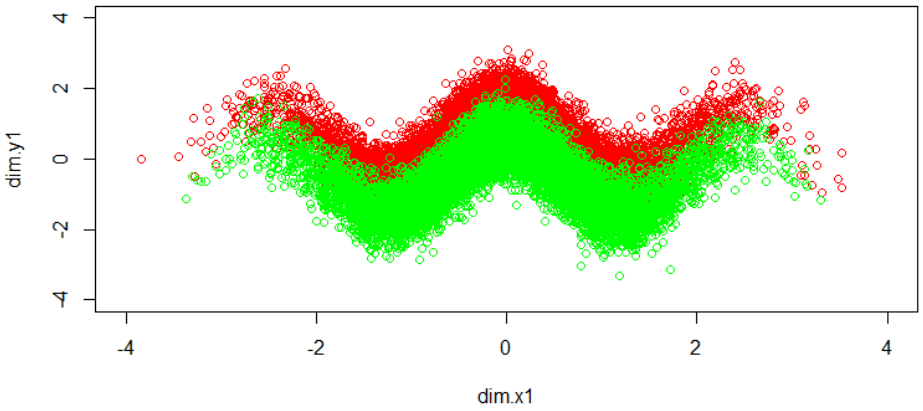

Figura 5.5: Dataset complex amb 1000 dades a cada classe

5.1.2 Descripció de les dades reals

En contrast amb les dades sintètiques, les dades reals obtingudes de diversos repositoris tenen un major nombre de variables i un nombre fixe d'elements. Per aquesta raó, no s'oferirà una descripció gràfica dels datasets, sinó una breu descripció de les seves variables.

En aquests datasets no s'ha realitzat cap mena de pre-processat excepte en aquells datasets amb valors nuls en alguna variable, on s'han eliminat els elements amb alguna variable amb valor nul.

El primer dataset utilitzat és el *Vertebral Column Pathologies* – VCP [34]. Aquest dataset consisteix en diverses mesures biomecàniques que s'han d'utilitzar per a predir si el pacient pateix alguna mena de malaltia de columna (*Abnormal*) o no (*Normal*). Aquest dataset té 310 mostres amb 6 atributs numèrics cadascuna.

El segon dataset utilitzat és el *Breast Cancer* – BC [35]. Aquest dataset consisteix en diverses mesures relacionades amb mostres de cèl·lules de possibles tumors que s'han d'utilitzar per a predir si la pacient pateix càncer de pit (*Malign*) o no (*Benign*). Aquest dataset té 683 mostres amb 9 atributs enters cadascuna.

El tercer dataset utilitzat és el *Pima Indian Diabetes* – PID [36]. Aquest dataset consisteix en diverses mesures biomèdiques que s'han d'utilitzar per a predir si el pacient pateix diabetis (*Diabetes*) o no (*Healty*). Aquest dataset té 768 mostres amb 8 atributs numèrics cadascuna.

El quart dataset utilitzat és el *Credit Card Fraud Detection* – CCFD [37]. Aquest dataset consisteix en el resultat d'una projecció PCA sobre un conjunt de dades bancàries confidencials que s'han d'utilitzar per a predir si l'usuari ha comés frau (*Fraud*) o no (*Legit*). Aquest dataset té 284,807 mostres amb 29 variables numèriques, i cal remarcar que és un dataset molt poc balancejat, ja que només el 0.172% dels elements del dataset pertanyen a la classe *Fraud*.

El quart dataset utilitzat és el *Letter Recognition* – LD [38]. Aquest dataset consisteix en diverses mesures numèriques extretes a partir d'un seguit d'imatges de diverses lletres majúscules de l'abecedari. El dataset original té etiquetades les 26 lletres de l'abecedari, de manera que les hem agrupat per a reduir el problema a determinar si una lletra és una vocal (*Vowel*) o una consonant (*Consonant*). Aquest dataset té 20000 mostres amb 16 atributs numèrics cadascuna.

5.2. Experiments per fases

Degut al gran nombre de possibles configuracions del mètode, no és possible realitzar una proba extensiva de totes les seves variants. Per aquesta raó, s'analitzaran per separat les diverses fases, utilitzant diferents criteris i conjunts de dades sintètics.

La idea és, al final d'aquesta secció, proposar (sempre que sigui possible) una tria predilecta per a la configuració de cada secció, tenint en compte els costos espacials, temporals i el rendiment de cada tècnica.

5.2.1 Fase de reducció

Com s'ha exposat a l'explicació del mètode, es poden utilitzar una gran varietat de mètodes per a la fase de reducció, tot i que la recomanació principal és utilitzar algoritmes de clustering. Aquí analitzarem aquells que han estat comentats en la secció de reducció de l'algoritme proposat.

Primer, avaluarem el rendiment teòric dels algoritmes de reducció. Com s'ha justificat anteriorment, podem utilitzar el coeficient de rendiment:

$$\eta_{red} = \|K - \bar{K}\|$$

On $\bar{K} = K_{x,c}K_c^+K_{c,x}$, K_c és la matriu de kernel $m \times m$ del conjunt de representants c_j, i $K_{x,c}$ és la matriu de kernel $n \times m$ de les dades x_i respecte els representants c_j.

A més, com a norma $\|\cdot\|$ utilitzarem la **norma de Frobenius**, que consisteix en computar la norma euclidiana codificant la matriu com un vector, per files o per columnes. La qualitat de la selecció és major quant menor sigui aquest coeficient.

Per tal d'estimar de manera adequada el valor d'aquest coeficient, per a cada tècnica es realitzarà un promig dels seus valors de 10 execucions, així com dels seus temps d'execució. Concretament, les dades següents s'han realitzat utilitzant conjunts de dades amb 1000 elements a cada classe, i triant sempre un conjunt amb 10 representants de cada classe (un 1% del total).

En vista als resultats obtinguts, podem apreciar que la majoria dels mètodes tenen un rendiment molt similar. Cal destacar que, tal com estava previst, tots els mètodes de clustering tenen un rendiment millor al de la Reducció Aleatòria, tot i que són més lents. Cal remarcar també que Spectral Clustering no té un rendiment molt superior als altres, tot i tenir un temps d'execució desproporcionadament gran.

Dataset	Kernel	Mètode	η_{red}	Temps
Lineal	Lineal	Reducció Aleatòria	$2.15 \cdot 10^{-12}$	2 ms
Lineal	Lineal	K-means	$1.86 \cdot 10^{-12}$	40 ms
Lineal	Lineal	K-medoids	$1.85 \cdot 10^{-12}$	5 seg
Lineal	Lineal	Spectral Clustering	$1.64 \cdot 10^{-12}$	1 min
Parabòlic	Quadràtic	Reducció Aleatòria	$1.46 \cdot 10^{-11}$	2 ms
Parabòlic	Quadràtic	K-means	$9.65 \cdot 10^{-12}$	40 ms
Parabòlic	Quadràtic	K-medoids	$1.79 \cdot 10^{-11}$	5 seg
Parabòlic	Quadràtic	Spectral Clustering	$1.80 \cdot 10^{-11}$	1 min
Complex	RBF($\sigma = 1$)	Reducció Aleatòria	4.02	2 ms
Complex	RBF($\sigma = 1$)	K-means	1.01	40 ms
Complex	RBF($\sigma = 1$)	K-medoids	0.28	5 seg
Complex	RBF($\sigma = 1$)	Spectral Clustering	1.84	1 min

Taula 5.1: Qualitat de les reduccions

Els resultats anteriors es poden generalitzar a diferent nombre de representants. A la gràfica següent es mostren els resultats de variar aquests paràmetres utilitzant només el dataset complex, que és el que ha mostrat una major variació del rendiment, probablement deguda a l'ús del kernel RBF.

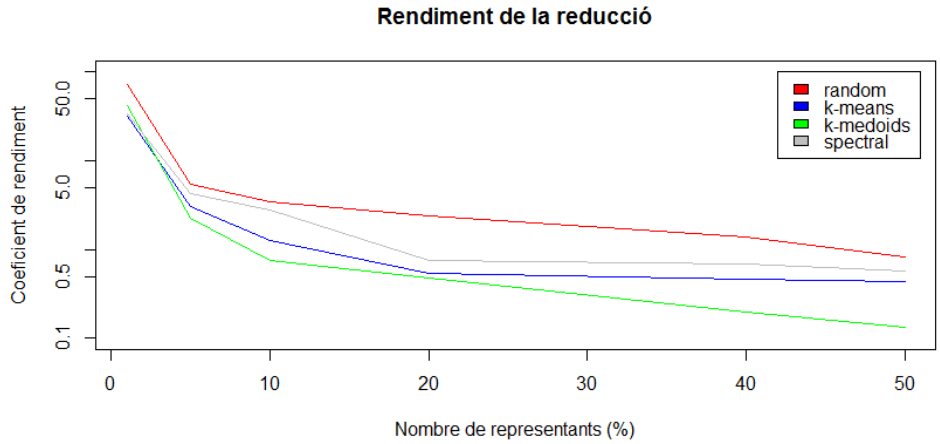

Figura 5.6: Rendiment per a diferents reduccions

A la gràfica anterior podem apreciar perfectament l'efecte del nombre de representants en la qualitat del subconjunt triat. Quant major és el nombre de representants triats, menor és el valor del coeficient de rendiment, fet que indica una millor qualitat del subconjunt. D'entre els mètodes examinats, podem observar que tots els mètodes de clustering tenen millor rendiment que la selecció aleatòria, tal com havíem suposat. A més, cal remarcar que entre els tres algoritmes utilitzats no hi ha una diferència significativa de rendiment, essent k-medoids lleugerament superior.

Ara, valorarem l'efecte de la tria del mètode de reducció en el resultat final l'algoritme de filtratge. Amb aquest objectiu, fixarem la mida i el nombre de mostres, utilitzant conjunts de dades amb 1000 elements a cada classe i 10 representants de cada classe (un 1% del total).

Un cop obtingudes les dades, les utilitzarem per entrenar una SVM i en mesurarem l'error generalitzat mitjançant un conjunt de dades independent. De cara al filtratge de les dades, s'ha utilitzat projecció mitjançant KFDA i selecció fixada al 20% de les dades d'entrada.

Dataset	Kernel	Mètode	Error (%)
Lineal	Lineal	–	8.10
Lineal	Lineal	Reducció Aleatòria	8.15
Lineal	Lineal	K-means	8.05
Lineal	Lineal	K-medoids	8.00
Lineal	Lineal	Spectral Clustering	8.10
Parabòlic	Quadràtic	–	5.85
Parabòlic	Quadràtic	Reducció Aleatòria	19.00
Parabòlic	Quadràtic	K-means	16.05
Parabòlic	Quadràtic	K-medoids	7.10
Parabòlic	Quadràtic	Spectral Clustering	14.80
Complex	RBF($\sigma = 1$)	–	21.05
Complex	RBF($\sigma = 1$)	Reducció Aleatòria	28.45
Complex	RBF($\sigma = 1$)	K-means	24.65
Complex	RBF($\sigma = 1$)	K-medoids	22.30
Complex	RBF($\sigma = 1$)	Spectral Clustering	22.50

Taula 5.2: Filtratge i entrenament amb diferents reduccions

Observant la taula anterior, podem arribar a la conclusió que el rendiment del mètode és força independent de l'algoritme de reducció triat, tot i que per norma general la selecció aleatòria sempre té un rendiment lleugerament inferior als altres. Destaca el bon funcionament de k-medoids en tots els casos, fins i tot millor que Spectral Clustering.

Per últim, mesurarem també la sensibilitat del rendiment dels diversos mètodes de reducció al nombre de representants triats. Igual que amb el gràfic anterior, utilitzarem el dataset complex, que és el que pot aportar resultats més rellevants.

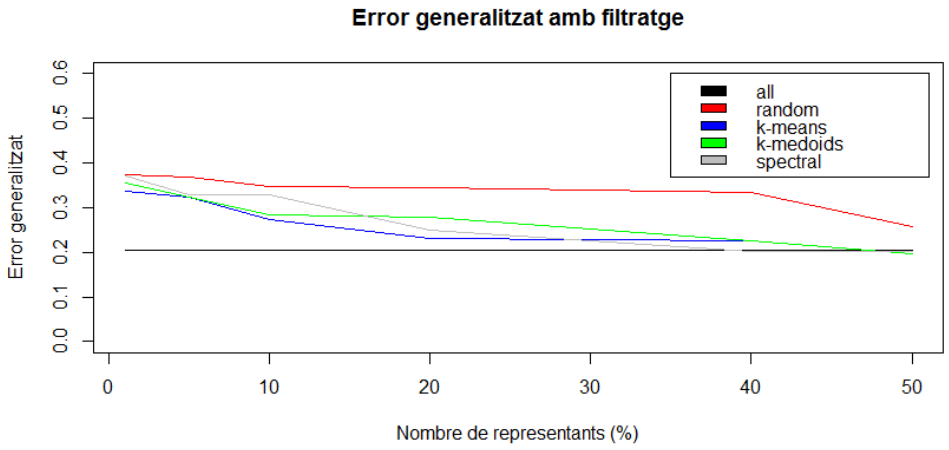

Figura 5.7: Error generalitzat per a diferents reduccions

Al gràfic anterior, podem apreciar que l'error resultant d'entrenar una SVM amb les dades filtrades pel mètode disminueix a mesura que incrementem el nombre de representants. Aquesta gràfica és coherent amb el que els coeficients de rendiment ens indicaven per als mètodes estudiats: Un major nombre de representants afavoreix una millor aproximació del vector *w*.

A més, podem observar que, en la majoria dels mètodes, n'hi ha prou amb utilitzar un 20% de les dades per a la projecció, ja que es comencen a obtenir aproximacions raonables. Cal recordar que l'experimentació anterior s'ha realitzat amb un kernel RBF, i que s'espera que els resultats siguin encara millors amb altres kernels.

També cal destacar que la selecció aleatòria no aconsegueix, en cap cas, apropar-se al rendiment dels altres mètodes de reducció, que tenen tots ells un rendiment similar.

En base als resultats obtinguts en aquest apartat, podem arribar a les conclusions següents:

1) En general, el criteri de reducció proposat és un bon indicador del rendiment del mètode de reducció, ja que és menor en aquells mètodes que, en utilitzar-los per al filtratge, assoleixen un menor error generalitzat el la SVM entrenada.

2) Tots els mètodes de reducció tenen un rendiment similar, i tots els mètodes de clustering ofereixen millors resultats que la selecció aleatòria.

3) D'entre els mètodes de reducció, la selecció aleatòria i k-means en són els que tenen un menor temps d'execució.

Per aquestes raons, es recomana utilitzar com a mètode estàndard de reducció l'algoritme de k-means. En cas que el nombre de dades sigui prou petit és recomanable la utilització de l'algoritme de k-medoids, que té un rendiment lleugerament superior a k-means.

5.2.2 Fase de projecció

Així com a la fase de reducció, a la fase de projecció també disposem de diversos mètodes que podem utilitzar. En aquesta secció analitzarem els que hem exposat en la secció de projecció de l'algoritme proposat.

En aquest cas, avaluarem conjuntament el rendiment teòric dels algoritmes de projecció i el seu rendiment pràctic. Com s'ha justificat anteriorment, per al primer podem utilitzar el coeficient de rendiment:

$$\eta_{prj} = \left|\left(\alpha^T K_{x,c}\beta\right)^2 - (\alpha^T K\alpha)(\beta^T K_c\beta)\right|$$

On K_c és la matriu de kernel $m \times m$ del conjunt de representants c_j, $K_{x,c}$ és la matriu de kernel $n \times m$ de les dades x_i respecte els representants c_j, α és l'expansió del vector w a estimar, i β és la expansió del nostre vector.

Per a l'execució dels experiments següents, s'han utilitzat conjunts de dades amb 10000 elements a cada classe. Per a la fase de reducció s'ha utilitzat l'algoritme de k-means amb 100 representants per a cada classe, i per a la fase de selecció s'ha limitat el conjunt de sortida a un 20% de les dades.

Dataset	Kernel	Mètode	η_{prj}	Error (%)	Temps
Lineal	Lineal	SVM	$1.90 \cdot 10^{-2}$	5.37	3 ms
Lineal	Lineal	KFDA	$1.37 \cdot 10^{-2}$	5.35	5 ms
Lineal	Lineal	KSDA	3.89	5.42	2 ms
Lineal	Lineal	Centroides	$6.66 \cdot 10^{-1}$	5.39	1 ms
Parabòlic	Quadràtic	SVM	$9.49 \cdot 10^{-3}$	5.31	3 ms
Parabòlic	Quadràtic	KFDA	$6.72 \cdot 10^{-3}$	5.28	7 ms
Parabòlic	Quadràtic	KSDA	$4.19 \cdot 10^{-1}$	14.79	3 ms
Parabòlic	Quadràtic	Centroides	$9.87 \cdot 10^{-2}$	5.36	2 ms
Complex	RBF($\sigma = 1$)	SVM	72.44	36.50	35 ms
Complex	RBF($\sigma = 1$)	KFDA	142.0	26.22	28 ms
Complex	RBF($\sigma = 1$)	KSDA	176.5	35.91	27 ms
Complex	RBF($\sigma = 1$)	Centroides	184.6	41.36	25 ms

Taula 5.3: Filtratge i entrenament amb diferents projeccions

En aquesta taula podem observar els diferents resultats obtinguts. Cal destacar que, tal i com estava previst, SVM i KFDA han resultat els mètodes més lents, mentre que KSDA i la projecció per centroides són significativament més ràpids. A nivell qualitatiu, és significatiu el fet que, en molts casos, ha donat millors resultats la projecció per centroides que KSDA. A més, la necessitat de validar el paràmetre de cost per a la projecció per SVM impedeix que aquesta obtingui millors resultats que KFDA.

En base als resultats obtinguts en aquest apartat, podem arribar a les conclusions següents:

1) En general, el criteri de reducció proposat és un bon indicador del rendiment del mètode de projecció, ja que és menor en aquells mètodes que, en utilitzar-los per al filtratge, assoleixen un menor error generalitzat el la SVM entrenada.

2) En general, els mètodes de projecció estudiats han resultat tenir el rendiment i temps d'execució esperats, en comparació amb els altres. La gran excepció és el rendiment de KSDA, que ha sigut, en molts casos, inferior a l'esperat.

3) D'entre els mètodes de projecció, el de la projecció per centroides és el més ràpid, mentre que KFDA és el que, en general, dona millors resultats..

Tot i el seu bon rendiment, la tria del paràmetre de cost a la projecció via SVM n'augmenta la complexitat, i no en fa una tria idònia. Dels altres mètodes, KFDA és el que assoleix un major rendiment, però també és més lent d'executar que els altres dos. Així doncs, la tria ha de ser determinada fonamentalment en base al nombre de representants triats: Si aquest nombre és raonable utilitzarem KFDA, i si és molt gran utilitzarem la projecció per centroides.

5.2.3 Fase de selecció

A diferència de les fases anteriors, per a la fase de selecció només es proposa un únic algoritme. Aquest, però, té dues possibles parametritzacions diferents (puresa o mida), i se n'ha d'estudiar el rendiment real segons aquests dos paràmetres.

Per a l'execució dels experiments següents, s'han utilitzat conjunts de dades amb 10000 elements a cada classe, i triant sempre un conjunt amb 100 representants de cada classe (un 1% del total). A més, per a la fase de reducció s'ha utilitzat k-means, i per a la fase de projecció s'ha utilitzat KFDA. Totes les dades s'han obtingut del conjunt parabòlic, de manera que s'ha utilitzat el kernel quadràtic per a la resolució.

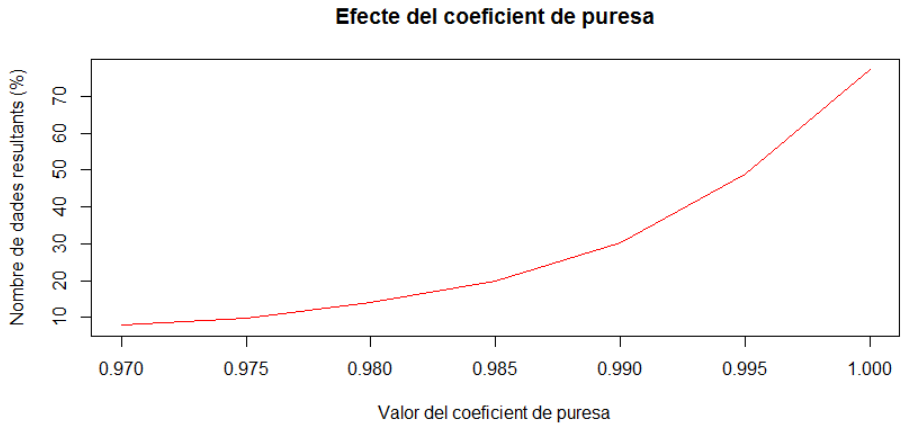

Figura 5.8: Nombre de dades resultants per a diferents valors de puresa

En aquests gràfics podem apreciar clarament l'efecte del coeficient de puresa: A mesura que aquest augmenta, s'incrementa també la mida del conjunt de dades filtrades, alhora que millora l'error generalitzat de la SVM entrenada amb aquestes dades. Cal remarcar, però, que per a conjunts molt senzills de separar (com és el cas dels datasets sintètics), el mètode és molt sensible al valor de p: Una petita variació provoca una gran diferència en la magnitud del nombre de vectors filtrats, i per tant, de l'error generalitzat de la SVM.

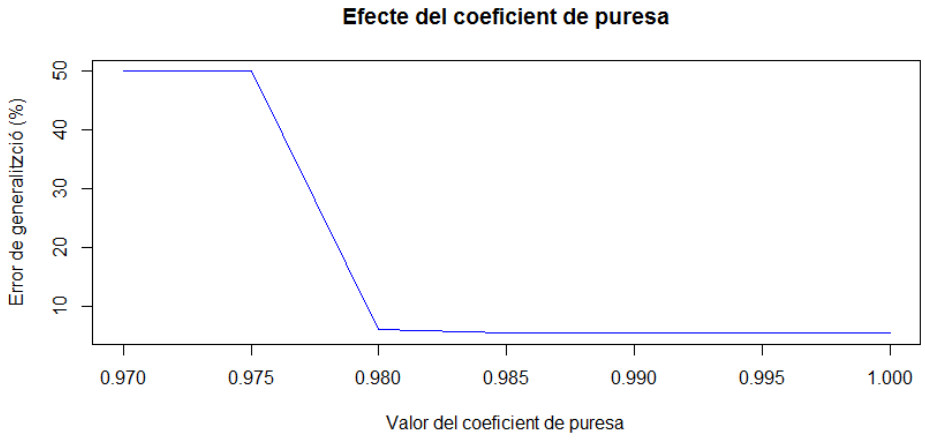

Figura 5.9: Error de generalització per a diferents valors de puresa

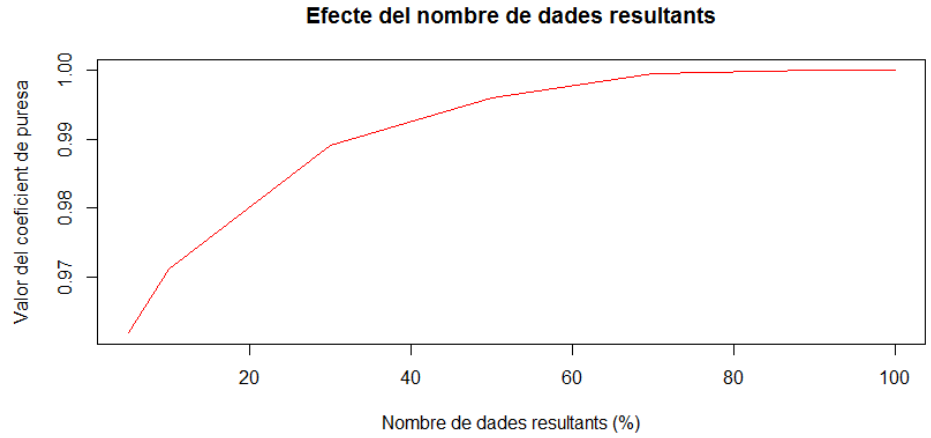

Figura 5.10: Coeficients de puresa per a diferent nombre de dades resultants

En aquests gràfics podem veure com la mida prefixada té un efecte similar al del coeficient de puresa: Com més gran és la mida del conjunt de dades filtrades, major és la puresa que es requereix per a obtenir-lo, i més millora l'error generalitzat de la SVM. Cal destacar, a més, que s'assoleix un punt en el que afegir més dades al conjunt de sortida ja no repercuteix en una millora de l'error de generalització: Així, l'objectiu de l'usuari de l'algoritme proposat ha de ser en tot moment trobar aquell punt que, amb el mínim nombre de dades, assoleixi un valor de l'error de generalització raonable.

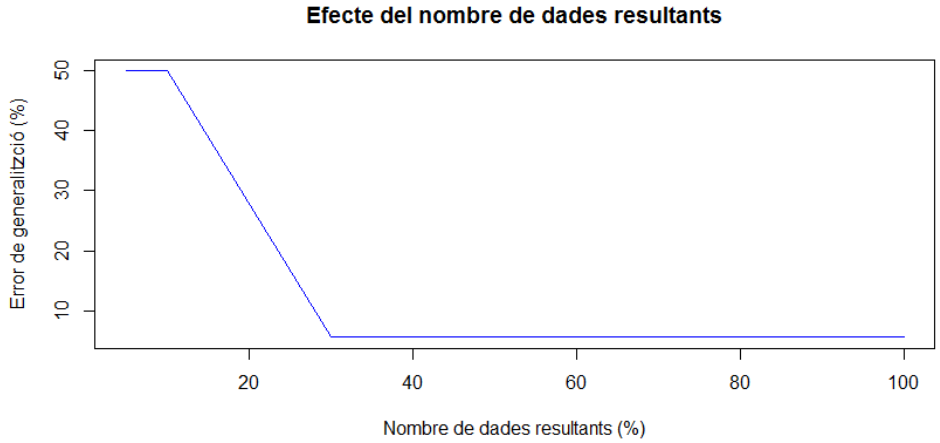

Figura 5.11: Error de generalització per a diferent nombre de dades resultants

5.2.4 Fase de construcció

Finalment, a la fase de construcció es generen tres subconjunts disjunts de dades, que es retornen com a resposta de l'algoritme. Aquests tres subconjunts es poden combinar a plaer, obtenint diferents resultats en la classificació de la SVM. Per aquesta raó, és interessant estudiar-ne totes les combinacions raonables, tot mesurant l'error generalitzat de les SVM entrenada amb cadascuna de les combinacions.

Per a l'execució dels experiments següents, s'han utilitzat conjunts de dades amb 10000 elements a cada classe. Per a la fase de reducció s'ha utilitzat l'algoritme de k-means amb 100 representants per a cada classe, i per a la fase de selecció s'ha limitat el conjunt de sortida a un 20% de les dades.

Dataset	Kernel	Subconjunt	Nº dades (%)	Error (%)
Lineal	Lineal	SV	20.00	7.73
Lineal	Lineal	SV + ADD	20.29	7.68
Lineal	Lineal	SV + OUT	21.93	9.25
Lineal	Lineal	SV + ADD + OUT	22.22	9.23
Parabòlic	Quadràtic	SV	20.00	5.62
Parabòlic	Quadràtic	SV + ADD	28.33	5.58
Parabòlic	Quadràtic	SV + OUT	21.06	50.00
Parabòlic	Quadràtic	SV + ADD + OUT	29.39	5.65
Complex	RBF($\sigma = 1$)	SV	20.00	25.38
Complex	RBF($\sigma = 1$)	SV + ADD	20.51	19.15
Complex	RBF($\sigma = 1$)	SV + OUT	29.50	73.44
Complex	RBF($\sigma = 1$)	SV + ADD + OUT	30.01	69.31

Taula 5.4: Filtratge i entrenament amb diferents projeccions

En vista als resultats de la taula anterior, podem concloure el següent:

1) Si afegim els punts que una SVM consideraria vectors suport, però que es troben a la banda contraria del marge (OUT), ens podem trobar casos en que això provoqui una interferència en l'entrenament de la nova SVM, ja que produeix un efecte similar al del soroll i en pot augmentar l'error generalitzat.

2) Si afegim el conjunt de punts addicionals per a balancejar la sortida (ADD), l'error generalitzat de la SVM entrenada tendeix a disminuir lleugerament. Cal valorar, en cada cas, si l'increment en el nombre de dades compensa la reducció de l'error.

3) Si afegim els dos conjunts alhora, l'error generalitzat té tendència a ser lleugerament menor que si afegim només el conjunt OUT, però en alguns casos perjudica greument la magnitud d'aquest error, fins al punt de generar hiperplans amb una orientació completament invertida (produint errors de generalització de més del 50% en alguns casos).

La conclusió d'aquest apartat, per tant, és la següent: La primera opció en tot entrenament ha de ser utilitzar només els conjunt de vectors suport retornats pel mètode. Si l'entrenament amb aquest conjunt de dades no dona resultats satisfactoris, afegim les dades de balanceig (ADD) per a equilibrar l'entrenament, i el repetim. En cap cas, però, hem d'utilitzar el conjunt de vectors suport al marge contrari (OUT), ja que en molts casos la seva utilització perjudica greument la nova SVM entrenada, i en cap cas ha mostrat una millora respecte la seva no utilització.

5.3. Experiments globals

Un cop analitzades les diferents fases de l'algoritme, ens trobem en condicions de decidir la configuració idònia per a cada dataset que se'ns pugui presentar. En general, les decisions per a la mida de la reducció i el mètode de projecció s'hauran de decidir a priori, mentre que el valor de puresa o la mida del conjunt de sortida es pot validar iterativament. A les taules es mostraran els resultats més significatius, i s'exposarà en profunditat aquella configuració que ha resultat millor.

Per als datasets sintètics, es visualitzaran diversos gràfics que permetran fer-se una idea visual del procés que s'està realitzant. A més, per a tots els datasets es visualitzaran les projeccions i el càlcul dels valors de puresa realitzats per l'algoritme.

En els casos en que s'utilitzi un kernel lineal, s'utilitzarà la versió lineal de l'algoritme, que permet assolir un menor temps d'execució per a obtenir el mateix resultat. En tots els casos, considerarem com a temps d'execució la suma del temps de l'algoritme de filtratge, més el temps d'entrenament de la SVM amb les dades resultants. El temps corresponent al classificador és el temps corresponent a l'algoritme de filtratge.

5.3.1 Dataset lineal

Per a les probes amb el dataset lineal s'utilitzarà una mostra de 50,000 dades de cada classe. Els conjunts de test i de validació són també de 50,000 dades de cada classe.

Kernel	Configuració	Nº dades (%)	Temps	Error (%)
Lineal	–	100.00	15 min	7.903
Lineal	FDA	20.00	2.5 min	7.909
Lineal	FDA	15.00	1.8 min	7.911
Lineal	Classificador (FDA)	100.00	4 seg	7.912

Taula 5.5: Filtratge i entrenament al dataset lineal

A continuació, exposarem els detalls de la configuració amb FDA (20% de les dades).

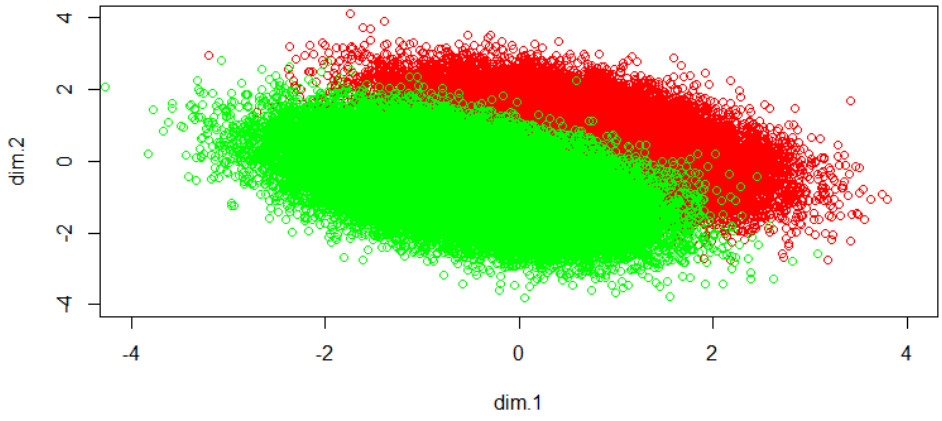

Figura 5.12: Dataset lineal utilitzat per a l'experimentació

El dataset utilitzat correspon al de la figura 5.12. La fase de preparació de les dades genera un conjunt de dades unidimensionals, de les quals en podem analitzar la distribució.

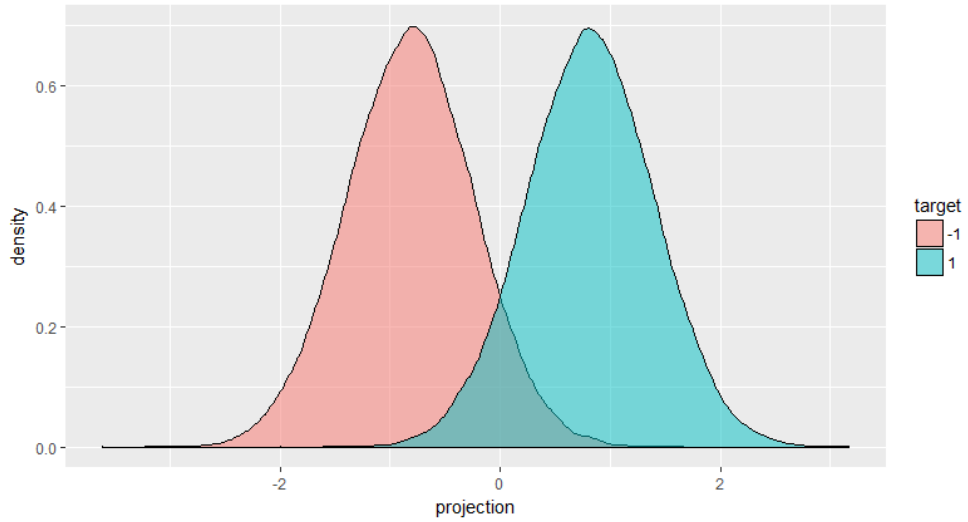

Figura 5.13: Projecció de les dades del dataset lineal

En el gràfic d'aquesta projecció podem veure clarament la normalitat de les dades projectades. Això permet afirmar que la direcció de projecció triada és adequada, i que el filtratge i la classificació posteriors seran raonablement bones.

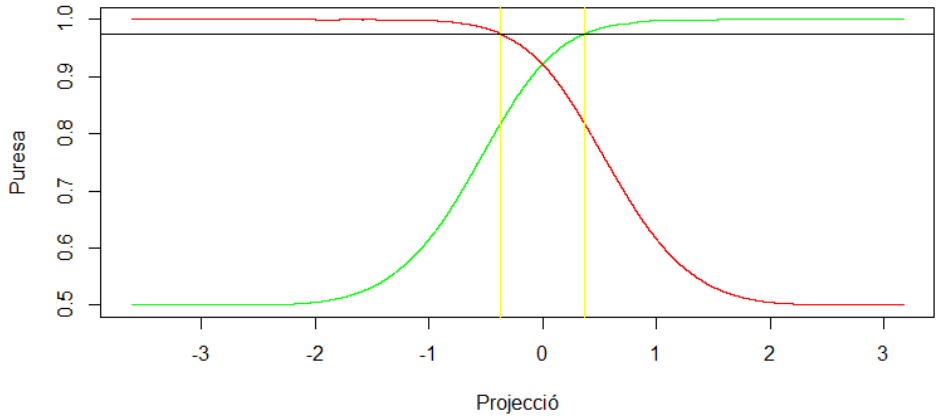

Figura 5.14: Puresa de les dades del dataset lineal

A la figura 5.14 podem veure els coeficients de puresa de cadascun dels punts de la projecció: En verd la puresa positiva, i en vermell la puresa negativa. La línia negra vertical representa el valor de puresa triat, i les dues línies grogues corresponen a $z_{i_{min}}$ i $z_{i_{max}}$. En aquesta gràfica es pot apreciar un comportament ideal, molt semblant al que prediuen els models teòrics.

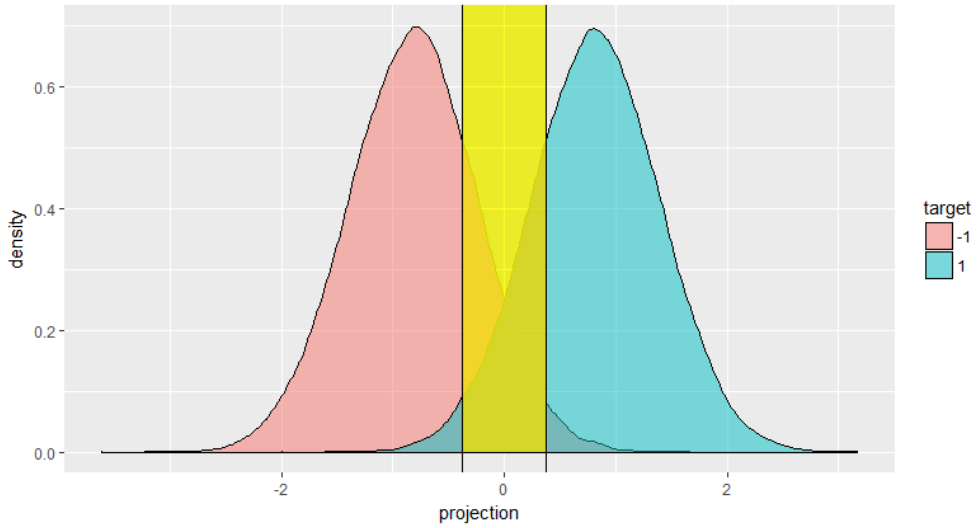

Figura 5.15: Selecció de les dades del dataset lineal

En el gràfic de la figura 5.15 podem apreciar la secció seleccionada de la projecció. Aquesta secció conté un 20% de les dades, i pràcticament tota la part de la intersecció de les distribucions de la projecció.

Per últim, podem realitzar una comparació entre els vectors suport que ha triat la SVM entrenada amb totes les dades, i els vectors que ha triat el nostre mètode de filtratge.

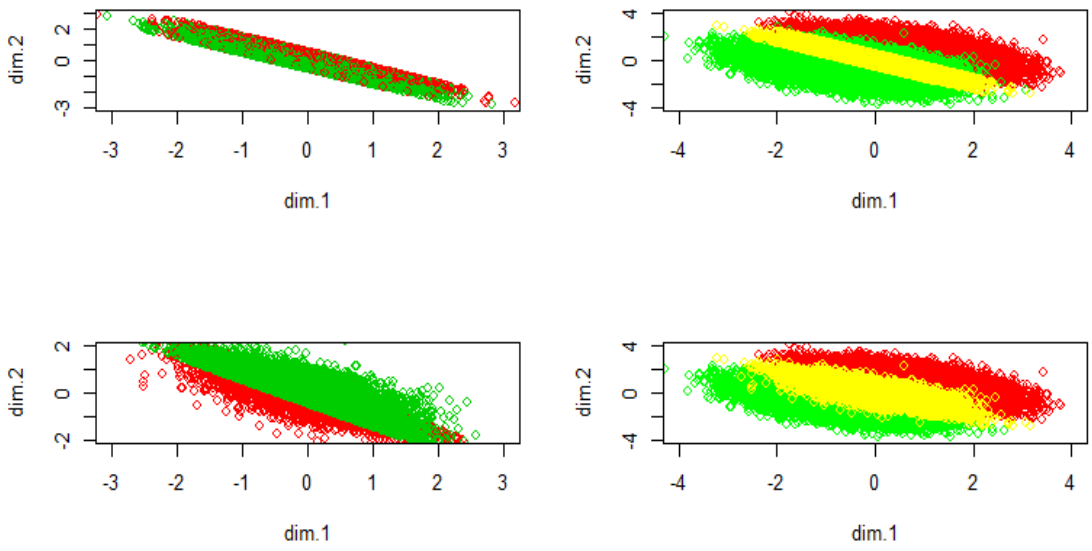

Figura 5.16: Comparació de vectors suport (dataset lineal)

A la figura 5.16 podem veure diverses imatges. A dalt a l'esquerra, podem veure el conjunt de dades seleccionades, i a dalt a la dreta aquest conjunt ressaltat en groc sobre les dades originals. A baix a l'esquerra podem veure el conjunt de vectors suport triat per la SVM entrenada amb les dades originals, i a baix a la dreta podem veure aquest conjunt ressaltat sobre les dades originals. Així, el nostre conjunt de dades seleccionades aproxima prou bé el conjunt de vectors suport, fins i tot eliminant dades innecessàries.

5.3.2 Dataset quasi separable

Per a les probes amb el dataset quasi separable s'utilitzarà una mostra de 50,000 dades de cada classe. Els conjunts de test i de validació són també de 50,000 dades de cada classe.

Kernel	Configuració	Nº dades (%)	Temps	Error (%)
Lineal	–	100.00	1.8 min	0.723
Lineal	FDA	10.00	20 seg	0.724
Lineal	FDA	1.00	8 seg	0.724
Lineal	Classificador (FDA)	100.00	3 seg	0.724

Taula 5.6: Filtratge i entrenament al dataset quasi separable

A continuació, exposarem els detalls de la configuració amb FDA (1% de les dades).

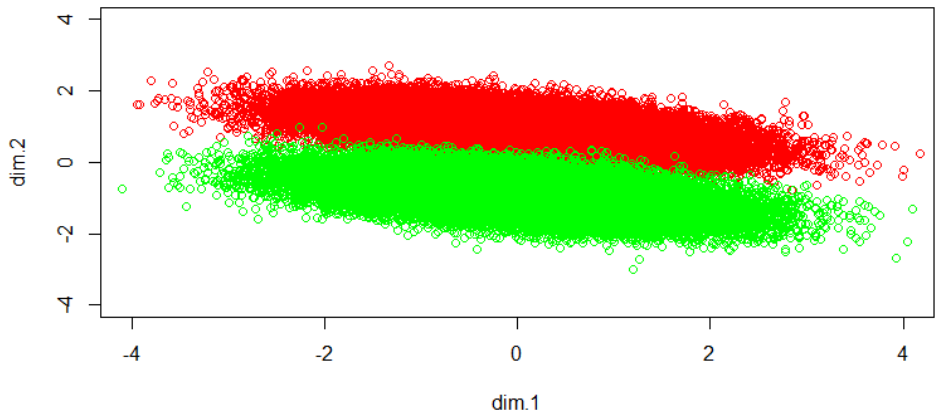

Figura 5.17: Dataset quasi separable utilitzat per a l'experimentació

El dataset utilitzat correspon al de la figura 5.17. La fase de preparació de les dades genera un conjunt de dades unidimensionals, de les quals en podem analitzar la distribució.

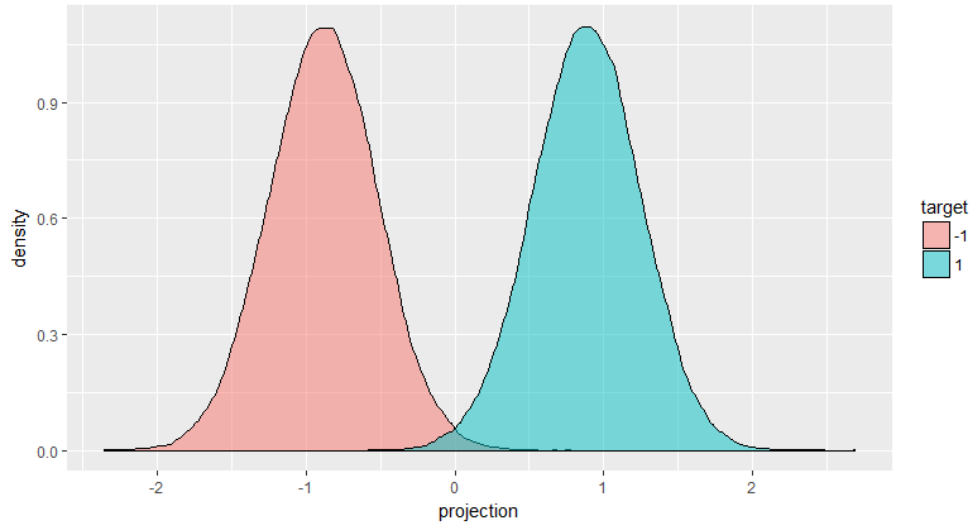

Figura 5.18: Projecció de les dades del dataset quasi separable

En el gràfic d'aquesta projecció podem veure clarament la normalitat de les dades projectades, igual que amb el dataset anterior. A més, podem veure que la seva intersecció és una secció molt més reduïda, amb una menor quantitat de dades. Això és un indicador que podem disminuir el nombre de vectors de sortida sense que això repercuteixi en una pèrdua de qualitat del classificador final.

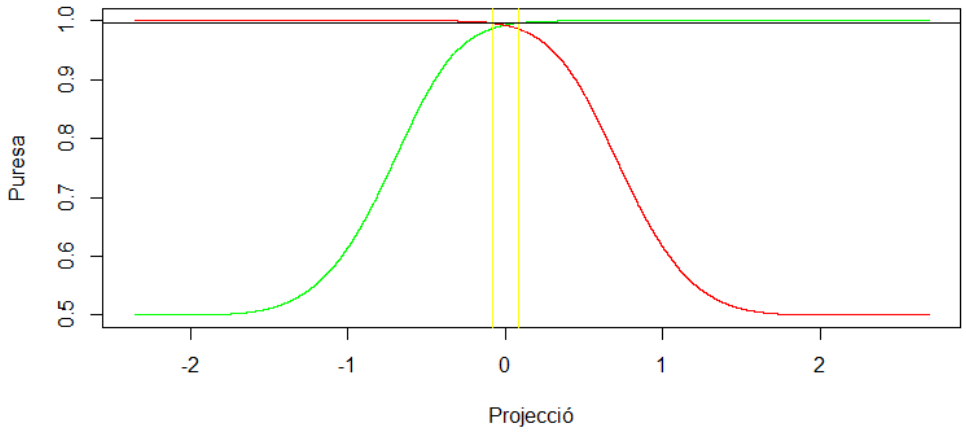

Figura 5.19: Puresa de les dades del dataset quasi separable

A la figura 5.19 podem veure els coeficients de puresa de cadascun dels punts de la projecció. Com que el conjunt és molt més separable que el dataset lineal, hi ha més punts amb una puresa del 100%, i la secció on ambdues pureses decauen és molt petita.

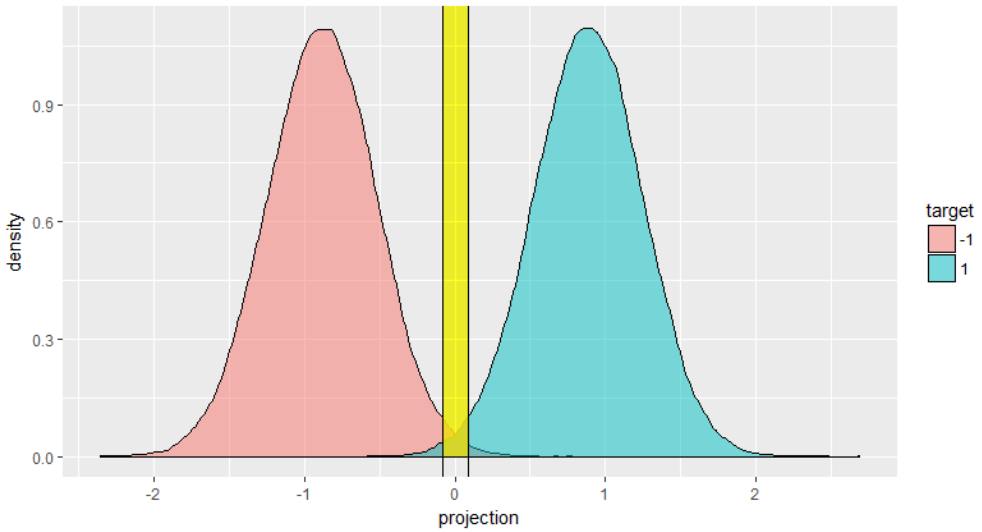

Figura 5.20: Selecció de les dades del dataset quasi separable

En el gràfic de la figura 5.20 podem apreciar la secció seleccionada de la projecció. Aquesta avarca una secció molt petita, degut al petit nombre de vectors de sortida.

Per últim, podem realitzar una comparació entre els vectors suport que ha triat la SVM entrenada amb totes les dades, i els vectors que ha triat el nostre mètode de filtratge.

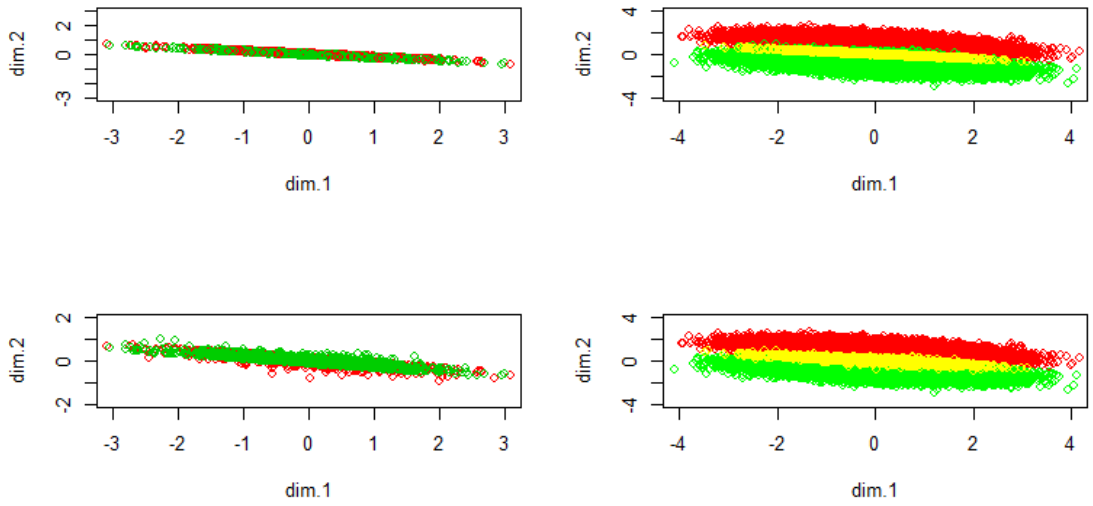

Figura 5.21: Comparació de vectors suport (dataset quasi separable)

A la figura 5.21 podem apreciar com el conjunt de dades de sortida és fins i tot menor al de vectors suport. Tot i això, el rendiment és pràcticament idèntic al de la SVM entrada amb totes les dades. La bona qualitat de l'aproximació es deu probablement al fet que la direcció *w* triada pel nostre mètode és molt semblant a la triada per la SVM.

5.3.3 Dataset separable

Per a les probes amb el dataset separable s'utilitzarà una mostra de 50,000 dades de cada classe. Els conjunts de test i de validació són també de 50,000 dades de cada classe.

Kernel	Configuració	Nº dades (%)	Temps	Error (%)
Lineal	–	100.00	6.4 seg	0.00
Lineal	FDA	1.00	3.9 seg	0.00
Lineal	Classificador (FDA)	100.00	3.1 seg	0.00

Taula 5.7: Filtratge i entrenament al dataset separable

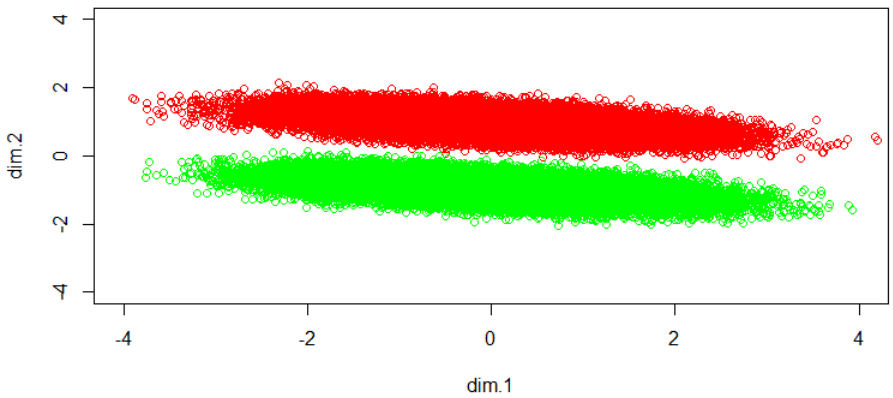

Figura 5.22: Dataset separable utilitzat per a l'experimentació

A continuació, exposarem els detalls de la configuració amb FDA (1% de les dades).

El dataset utilitzat correspon al de la figura 5.22. La fase de preparació de les dades genera un conjunt de dades unidimensionals, de les quals en podem analitzar la distribució.

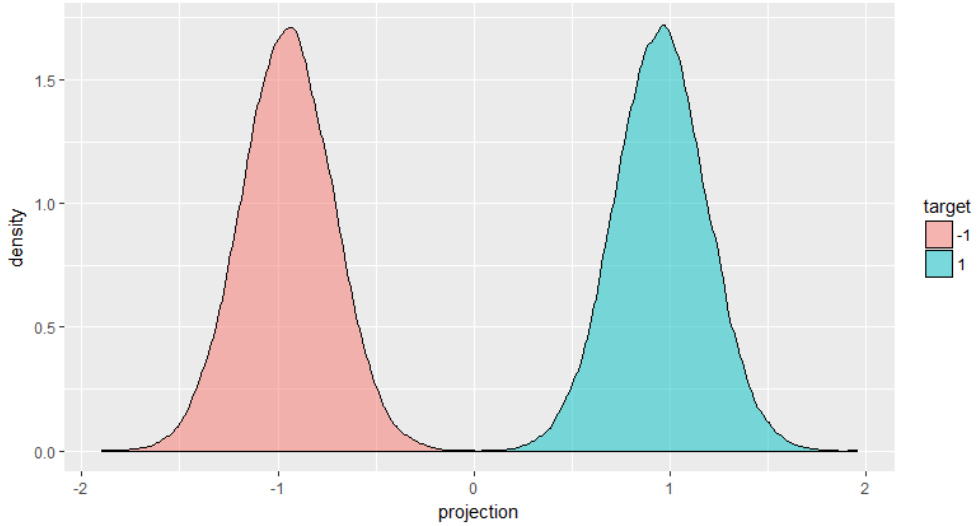

Figura 5.23: Projecció de les dades del dataset quasi separable

En el gràfic d'aquesta projecció podem veure clarament la normalitat de les dades projectades, igual que en els altres datasets. A més, podem veure que la seva intersecció és pràcticament nul·la, ja que en el sentit de la projecció les dades no arriben a creuar-se mai. Això permet agafar una quantitat arbitràriament petita de dades per a realitzar la classificació sense que això suposi una pèrdua de qualitat del classificador final.

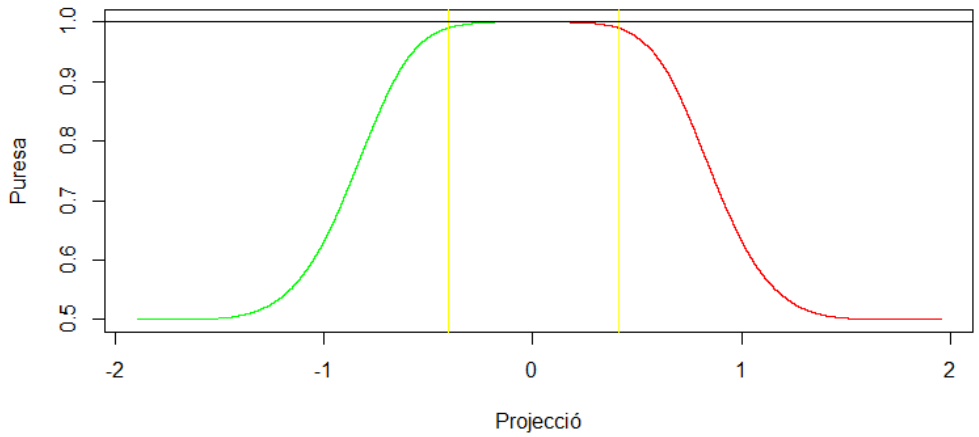

Figura 5.24: Puresa de les dades del dataset separable

A la figura 5.24 podem veure els coeficients de puresa de cadascun dels punts de la projecció. Com que el conjunt és separable, tots els punts tenen, o bé una puresa positiva del 100%, o una puresa negativa del 100%.

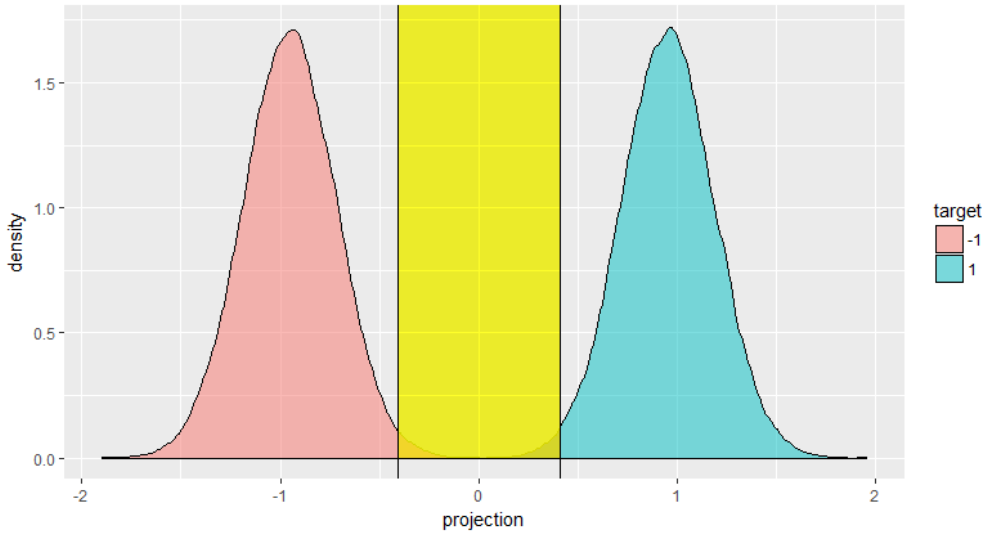

Figura 5.25: Selecció de les dades del dataset separable

En el gràfic de la figura 5.25 podem apreciar la secció seleccionada de la projecció. Aquesta avarca una secció més gran, per tal de poder englobar el nombre de dades desitjat.

Per últim, podem realitzar una comparació entre els vectors suport que ha triat la SVM entrenada amb totes les dades, i els vectors que ha triat el nostre mètode de filtratge.

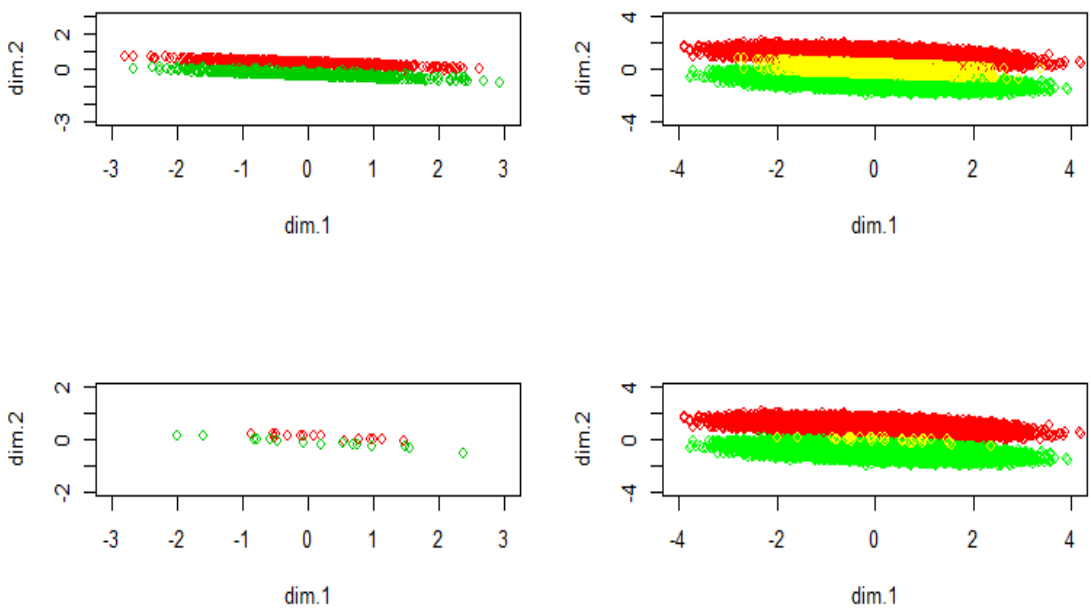

Figura 5.26: Comparació de vectors suport (dataset separable)

En aquesta figura podem apreciar que el conjunt de dades de sortida excedeix el conjunt de vectors suport. Això ens indica, per tant, que podríem haver reduït encara més la mida del conjunt de sortida.

5.3.4 Dataset parabòlic

Per a les probes amb el dataset parabòlic s'utilitzarà una mostra de 50,000 dades de cada classe. Els conjunts de test i de validació són també de 50,000 dades de cada classe.

Kernel	Configuració	Nº dades (%)	Temps	Error (%)
Quadràtic	–	100.00	18 min	5.378
Quadràtic	KFDA	20.00	3 min	5.577
Quadràtic	KFDA + ADD	15.00 / 21.85	3 min	5.420
Quadràtic	Classificador (KFDA)	100.00	27 seg	5.440
Quadràtic	SVM ($C = 2^2$)	20.00	3 min	5.415
Quadràtic	SVM ($C = 2^2$) + ADD	15.00 / 17.18	3 min	5.407
Quadràtic	Classificador (SVM, $C = 2^2$)	100.00	27 seg	5.542

Taula 5.8: Filtratge i entrenament al dataset parabòlic

En tots els casos, s'ha utilitzat l'algoritme de k-means per a la fase de reducció, utilitzant 50 representants de cada classe (un 0.1%). Per a estimar la direcció de projecció, s'ha utilitzat el promig de 10 projeccions diferents obtingudes amb la configuració anterior.

En els casos en que s'utilitza KFDA, l'algoritme utilitzat per al classificador ha estat LDA. En els casos en que s'utilitza SVM, per al classificador s'ha utilitzat SVM1D amb el mateix valor de C utilitzat a la projecció. Per a triar el valor de C, s'han realitzat diferents execucions i s'ha triat aquell valor que maximitza l'error generalitzat del classificador que retorna el mètode.

A la taula anterior, podem apreciar clarament l'efecte positiu del conjunt addicional: Si utilitzem un 20% de les dades obtenim un resultat pitjor que si n'utilitzem un 15% i un 5% de dades addicionals per a balancejar. El millor resultat s'ha obtingut utilitzant projecció per SVM, però aquesta té l'inconvenient de requerir la validació del paràmetre de cost. Si disposem de prou temps de computació és interessant validar aquesta opció, però si no es disposa de prou temps és preferible la utilització de la projecció per KFDA, que no requereix de la validació de cap mena de paràmetre.

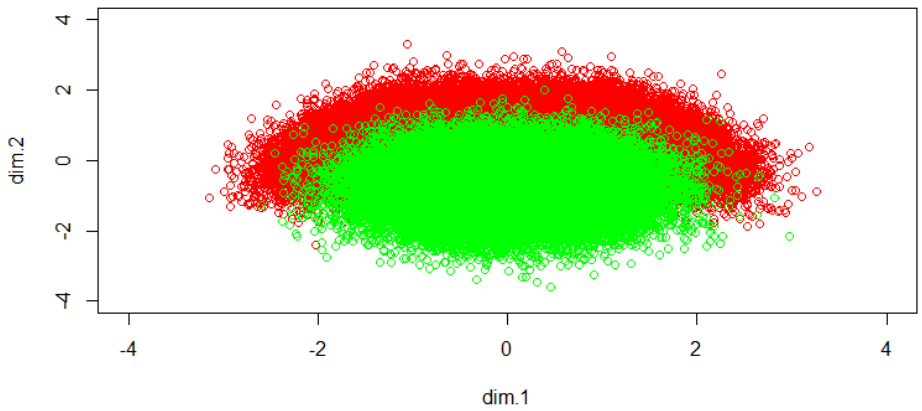

Figura 5.27: Dataset parabòlic utilitzat per a l'experimentació

A continuació, exposarem els detalls de la configuració amb KFDA i el conjunt de dades addicional per a balancejar.

El dataset utilitzat correspon al de la figura 5.27. La fase de preparació de les dades genera una reducció de les dades i un conjunt de dades unidimensionals. Ambdós conjunts els podem visualitzar en un gràfic:

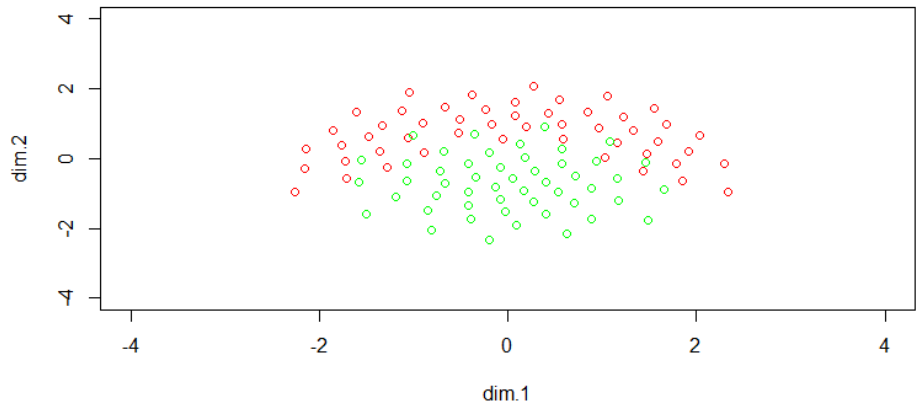

Figura 5.28: Exemple de reducció de les dades del dataset parabòlic

A la figura 5.28 podem visualitzar una de les reduccions utilitzades. Aquestes semblen representar satisfactòriament l'estructura del dataset a filtrar.

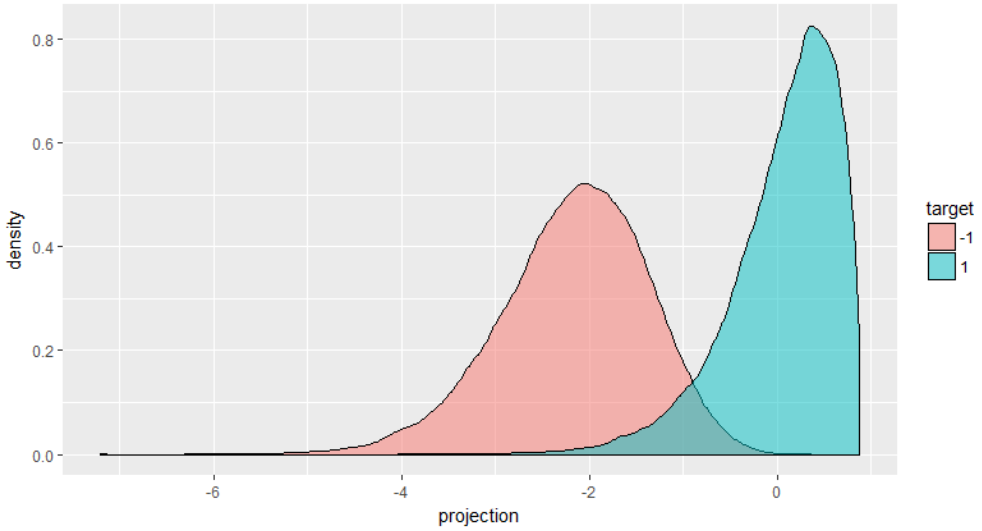

Figura 5.29: Projecció de les dades del dataset parabòlic

En el gràfic d'aquesta projecció podem veure que la projecció de les dades segueix mantenint una distribució normal, però que aquestes ja no comparteixen la mateixa variància. Tot i això, la projecció sembla permetre una separació prou bona de les dades.

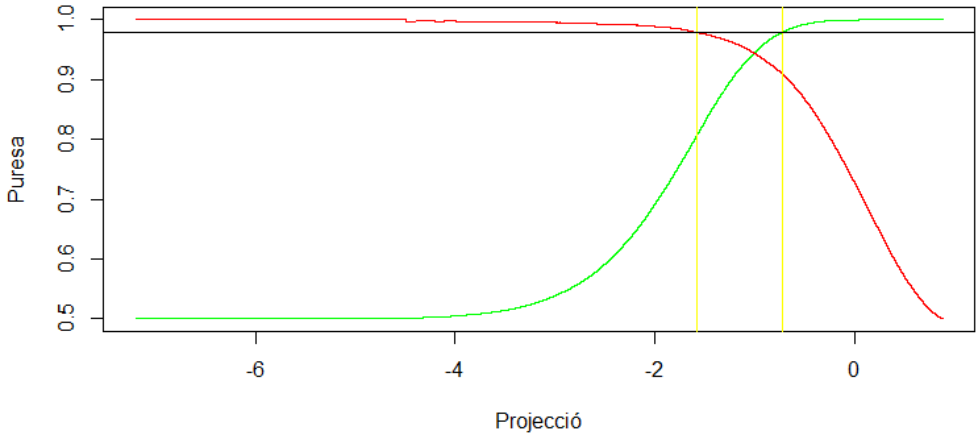

Figura 5.30: Puresa de les dades del dataset parabòlic

A la figura 5.30 podem veure els coeficients de puresa de cadascun dels punts de la projecció. Podem apreciar com hi ha un major nombre de dades de la classe negativa amb una puresa propera al 100% que no pas de la classe positiva. La selecció, però, té una amplitud similar a la obtinguda al dataset lineal.

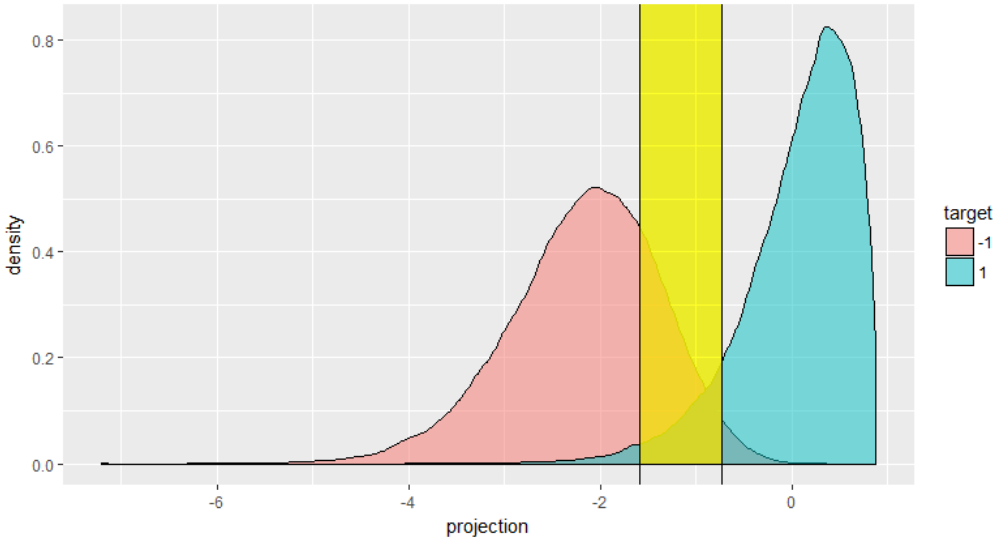

Figura 5.31: Selecció de les dades del dataset parabòlic

En el gràfic de la figura 5.31 podem apreciar la secció seleccionada de la projecció. Aquesta avarca més elements de la classe negativa que de la positiva, degut a que les dades es solapen menys en aquella àrea.

Per últim, podem realitzar una comparació entre els vectors suport que ha triat la SVM entrenada amb totes les dades, i els vectors que ha triat el nostre mètode de filtratge.

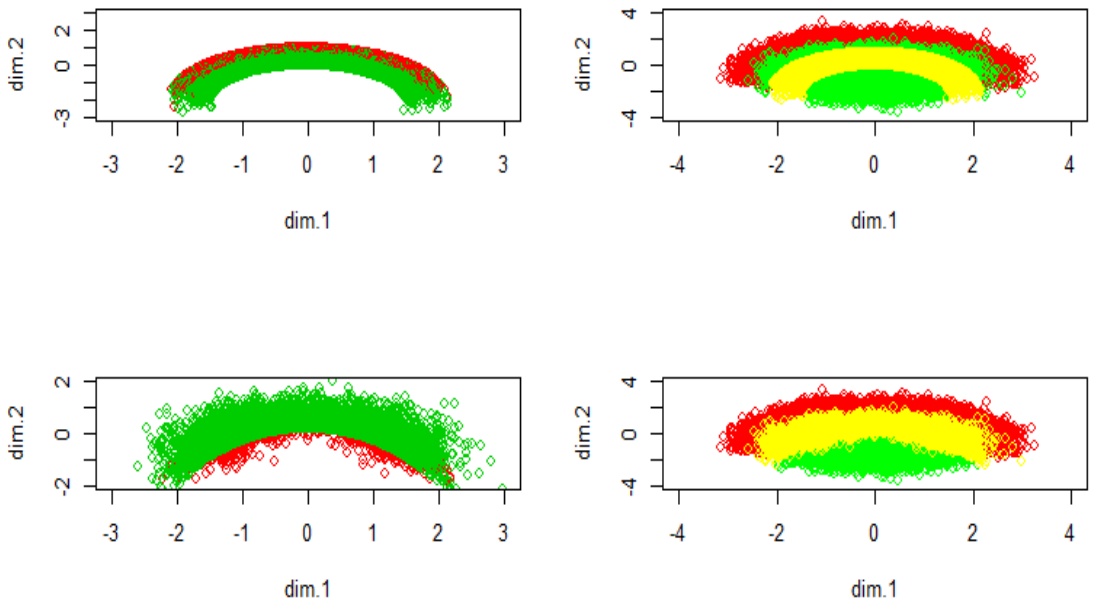

Figura 5.32: Comparació de vectors suport (dataset parabòlic)

En aquesta figura podem veure que el conjunt de dades de sortida selecciona molts més elements de la classe positiva que al conjunt de vectors. Això és degut a que aquest excedent de dades prové del conjunt de dades addicional, que ajuden a millorar la qualitat de la classificació final encara que no siguin seleccionats com a vectors suport.

5.3.5 Dataset complex

Per a les probes amb el dataset complex s'utilitzarà una mostra de 50,000 dades de cada classe. Els conjunts de test i de validació són també de 50,000 dades de cada classe.

Kernel	Configuració	Nº dades (%)	Temps	Error (%)
RBF($\gamma = 8$)	–	100.00	3 h	16.06
RBF($\gamma = 8$)	KFDA	40.00	55 min	16.39
RBF($\gamma = 8$)	Classificador (KFDA)	100.00	20 seg	18.75
RBF($\gamma = 8$)	SVM ($C = 2^2$)	40.00	40 min	17.37
RBF($\gamma = 8$)	Classificador (SVM, $C = 2^2$)	100.00	20 seg	16.59

Taula 5.9: Filtratge i entrenament al dataset complex

En tots els casos, s'ha utilitzat l'algoritme de k-means per a la fase de reducció, utilitzant 500 representants de cada classe (un 1%). Per a estimar la direcció de projecció, s'ha utilitzat el promig de 10 projeccions diferents obtingudes amb la configuració anterior.

El nombre de representants de cada classe s'ha incrementat degut a l'elevada complexitat del kernel utilitzat. Si el nombre de representants és massa petit, la projecció degenera i no és possible realitzar una estimació adequada de la direcció de projecció w.

La resta de criteris i paràmetres de configuració són els mateixos que els utilitzats per a treballar amb el dataset parabòlic.

A continuació, exposarem els detalls de la configuració amb KFDA i el conjunt de dades addicional per a balancejar.

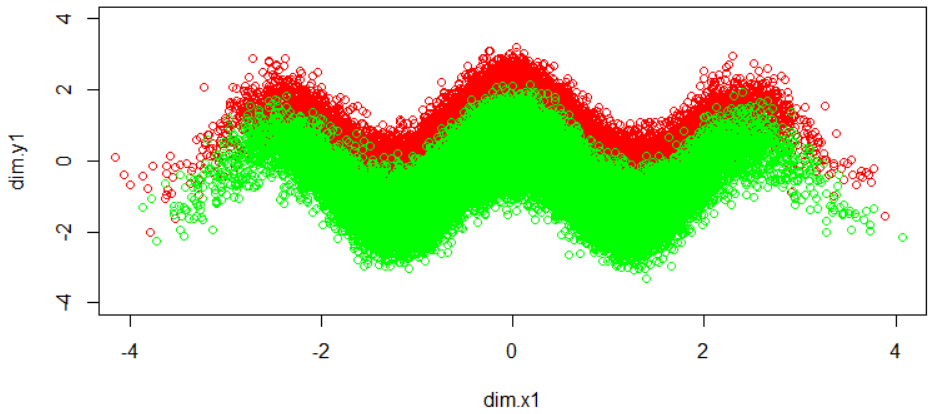

Figura 5.33: Dataset complex utilitzat per a l'experimentació

El dataset utilitzat correspon al de la figura 5.33. La fase de preparació de les dades genera una reducció de les dades i un conjunt de dades unidimensionals. Ambdós conjunts els podem visualitzar en un gràfic:

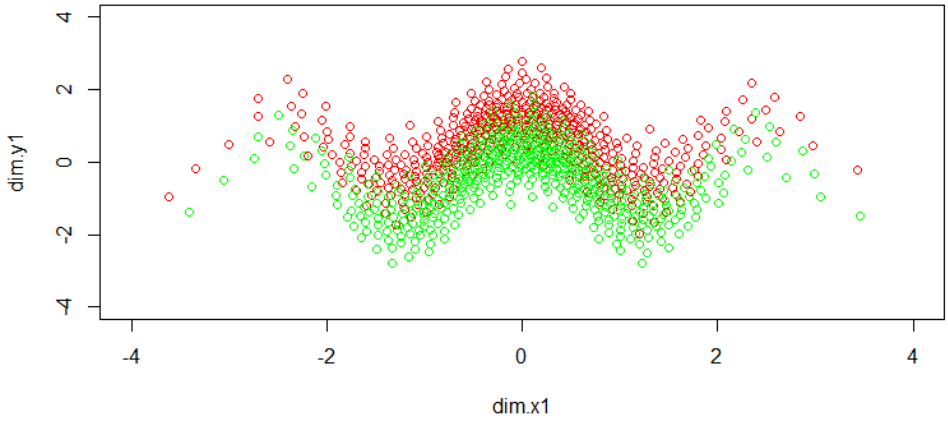

Figura 5.34: Exemple de reducció de les dades del dataset complex

A la figura 5.34 podem visualitzar una de les reduccions utilitzades. Aquestes semblen representar satisfactòriament l'estructura del dataset a filtrar.

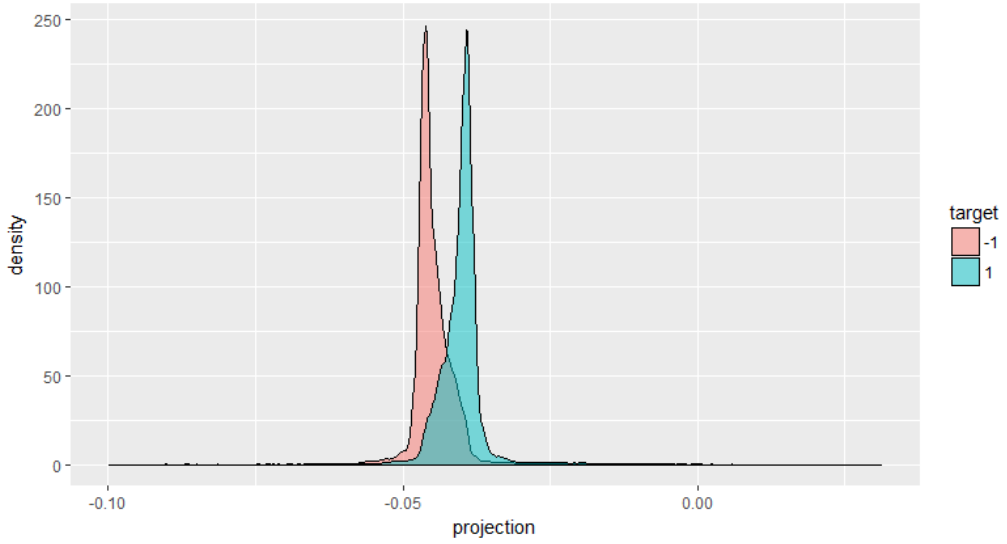

Figura 5.35: Projecció de les dades del dataset complex

En el gràfic d'aquesta projecció podem veure que la normalitat de la projecció no és tan clara com en els altres datasets. Això és degut a l'elevada dimensionalitat del Feature Space generat pel kernel RBF, que provoca una complexitat en les dades que impedeix una adequada aproximació del vector de projecció *w*. Tot i això, la projecció sembla prou bona per a realitzar el filtratge de les dades.

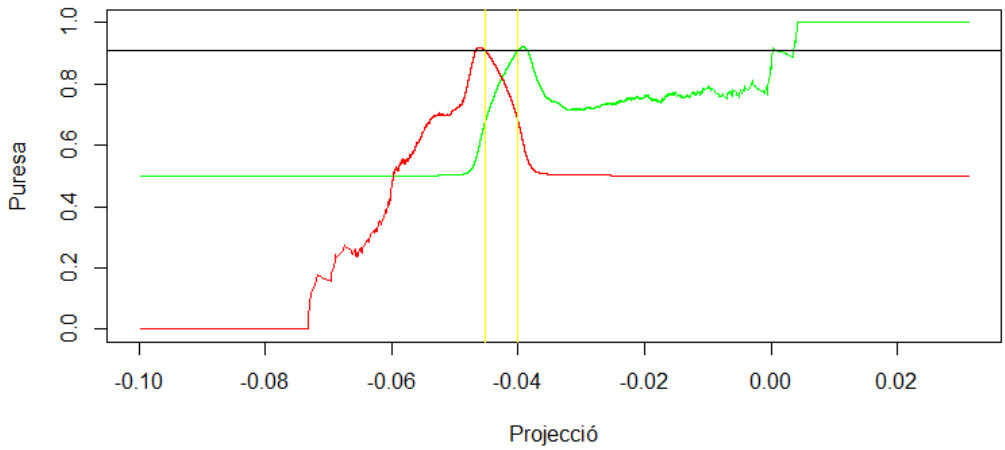

Figura 5.36: Puresa de les dades del dataset complex

A la figura 5.36 podem veure els coeficients de puresa de cadascun dels punts de la projecció. En aquesta gràfica podem apreciar la gran irregularitat de la distribució de les dades, que repercuteix en uns valors de puresa també molt irregulars. La zona seleccionada, però, té una distribució més semblant a la dels altres datasets, de manera que s'espera que la qualitat del conjunt seleccionat sigui també similar als casos anteriors.

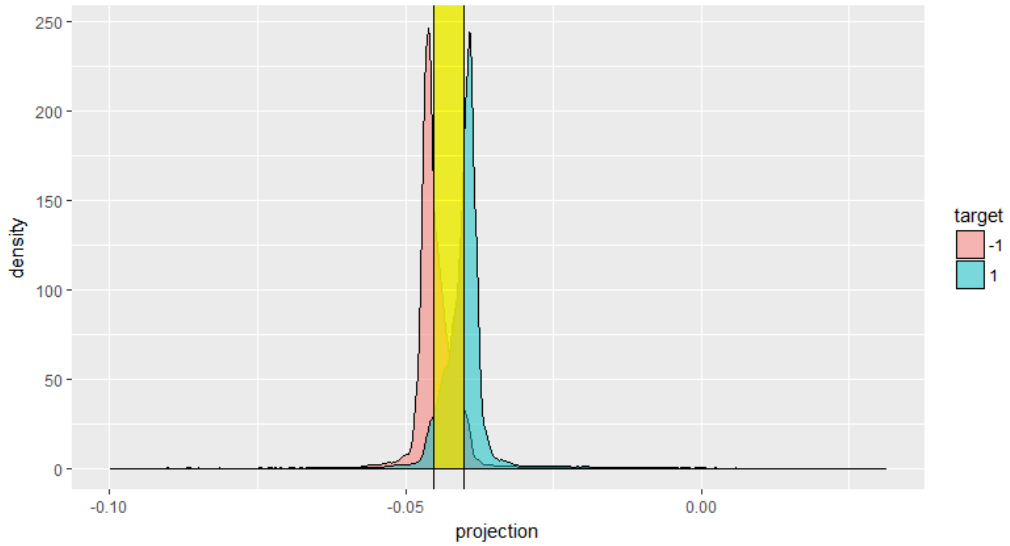

Figura 5.37: Selecció de les dades del dataset complex

En el gràfic de la figura 5.37 podem apreciar la secció seleccionada de la projecció. Aquesta és molt estreta deguda a la gran concentració de dades en els valors centrals, tot i que correspon a un percentatge molt important de les dades (un 40%).

Per últim, podem realitzar una comparació entre els vectors suport que ha triat la SVM entrenada amb totes les dades, i els vectors que ha triat el nostre mètode de filtratge.

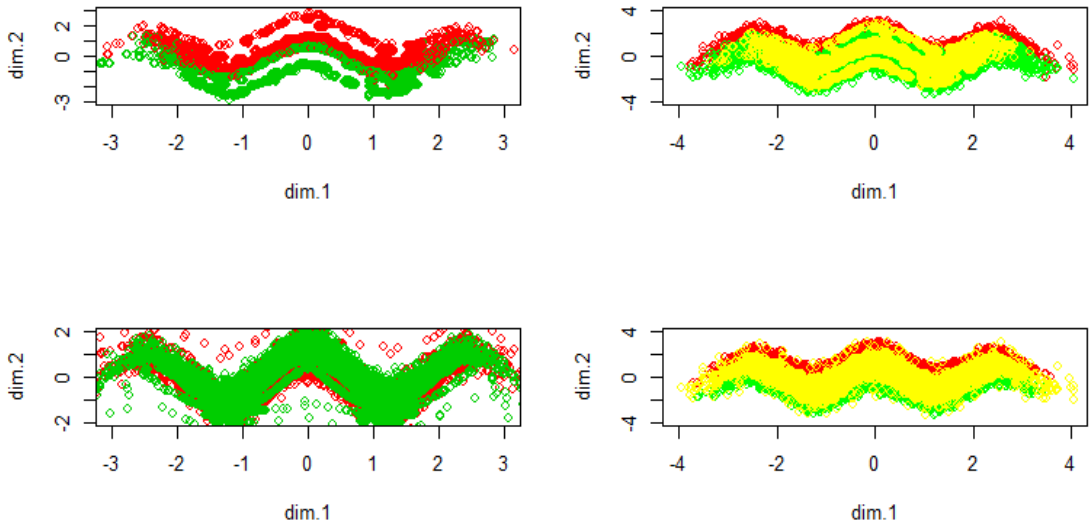

Figura 5.38: Comparació de vectors suport (dataset complex)

A la figura 5.38 es pot apreciar perfectament la feina del nostre algoritme: Només s'han retornat com a dades resultants aquelles més properes a la separació de les dades, eliminant una quantitat important de dades a les zones més pures. Així, tot i que el dataset requereix que s'utilitzin més dades per a obtenir un error raonable i s'hagin utilitzat molts més representants que en els altres datasets, els resultats han estat igualment satisfactoris.

5.3.6 Dataset VCP

Per a realitzar les probes amb el dataset VCP, s'ha realitzat una partició de les dades de tal manera que el 60% pertanyen ara al conjunt d'entrenament, el 20% al conjunt de test i el 20% restant al de validació. A més, per tal de poder realitzar una comparació amb els resultats del mètode *opposite maps*, s'utilitzaran els mateixos paràmetres que en els seus experiments en la configuració dels kernel. El paràmetre de cost, però, s'ha hagut de validar independentment.

Com que estem treballant amb un conjunt de dades prou reduït, no realitzarem fase de projecció en cap del casos. Per començar, aplicarem la versió lineal del mètode, amb una pures del 100%, per d'obtenir una primera impressió del dataset.

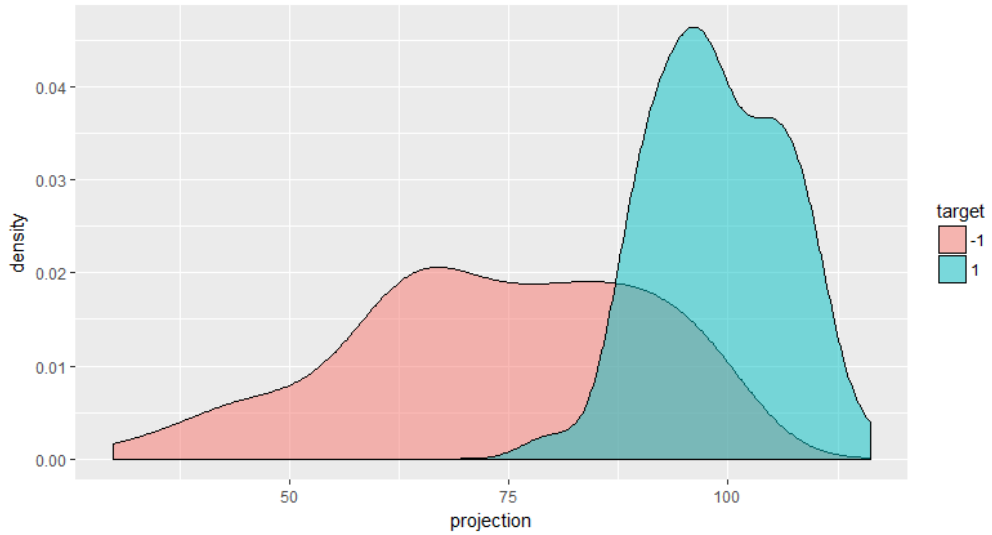

Figura 5.39: Projecció lineal de les dades (dataset VCP)

La projecció de les dades no és molt irregular, tot i que no es troba idènticament distribuïda. A continuació, seleccionem les dades utilitzant un 100% de puresa.

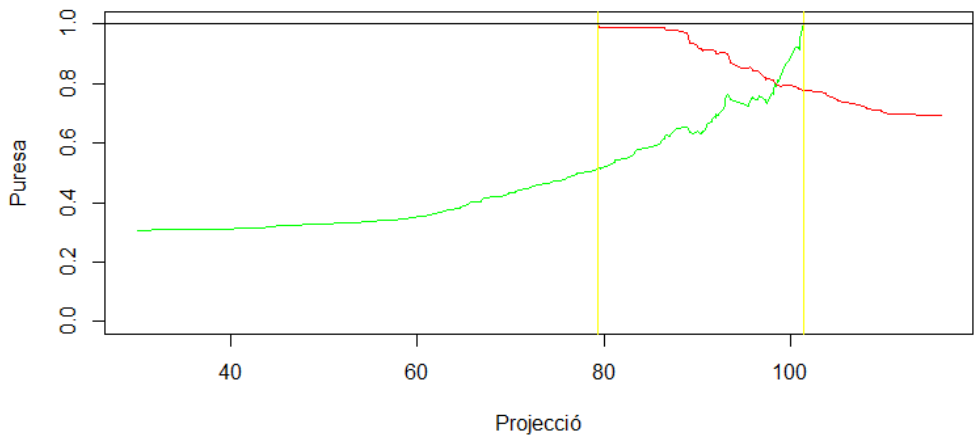

Figura 5.40: Puresa de les dades (dataset VCP)

A la figura 5.40 podem apreciar que la puresa de les dades és bastant regular al llarg de la projecció realitzada. A més, la tria de puresa del 100% ja permet assolir una reducció del 50% de les dades.

Ara, entrenem una SVM amb el conjunt de dades filtrades. Els resultats es mostren a la taula 5.10, al final de l'apartat.

Ara, realitzarem una aproximació no lineal al dataset. Per a poder realitzar una comparació adequada, utilitzarem la mateixa configuració que Opposite Maps per al kernel RBF ($\sigma = 1.5$).

Si executem el nostre mètode amb el kernel especificat, el conjunt d'entrenament resulta separable linealment en Feature Space. Per aquesta raó, fixarem el nombre de dades de filtratge en un 50% de les dades originals.

Kernel	Mètode	Nº dades (%)	Error (%)
Lineal	–	100.00	15.1
Lineal	Opposite Maps	82.50	14.4
Lineal	Filtratge per puresa (projecció FDA)	58.66	15.1
RBF($\sigma = 1.5$)	–	100.00	14.7
RBF($\sigma = 1.5$)	Opposite Maps	80.30	15.4
RBF($\sigma = 1.5$)	Filtratge per puresa (projecció KFDA)	50.00	15.2

Taula 5.10: Resultats del dataset VCP

En general, els resultats són millors que els de Opposite Maps tant a nivell de reducció com d'error, ja que el gran control del filtratge per puresa sobre la mida del conjunt de sortida permet oferir reduccions molt més grans sense patir una pèrdua de qualitat de la classificació. Cal remarcar, però, que el conjunt utilitzat és molt petit (només 310 mostres), i és possible que els resultats siguin fortament dependents dels subconjunt s utilitzats per a entrenament, validació i test.

5.3.7 Dataset BC

Per a realitzar les probes amb el dataset BC, s'ha realitzat una partició de les dades de tal manera que el 60% pertanyen ara al conjunt d'entrenament, el 20% al conjunt de test i el 20% restant al de validació. A més, per tal de poder realitzar una comparació amb els resultats del mètode *opposite maps*, s'utilitzaran els mateixos paràmetres que en els seus experiments en la configuració dels kernel. El paràmetre de cost, però, s'ha hagut de validar independentment.

Com que estem treballant amb un conjunt de dades prou reduït, no realitzarem fase de projecció en cap del casos. Per començar, aplicarem la versió lineal del mètode, amb una pures del 100% per d'obtenir una primera impressió del dataset.

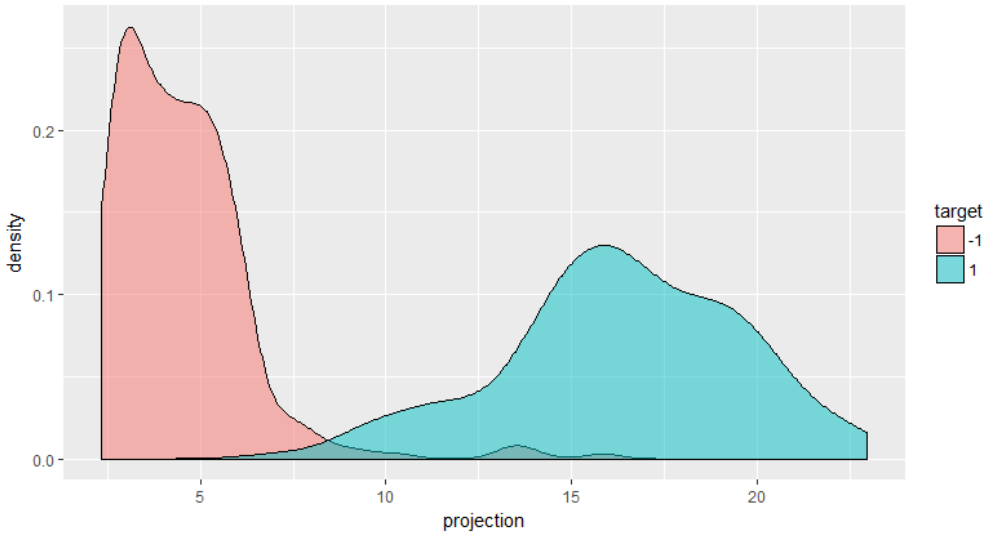

Figura 5.41: Projecció lineal de les dades (dataset BC)

En aquest cas, la projecció de les dades si que és una mica més irregular, però encara es poden distingir perfectament les dues classes. A continuació, seleccionem les dades utilitzant un 100% de puresa.

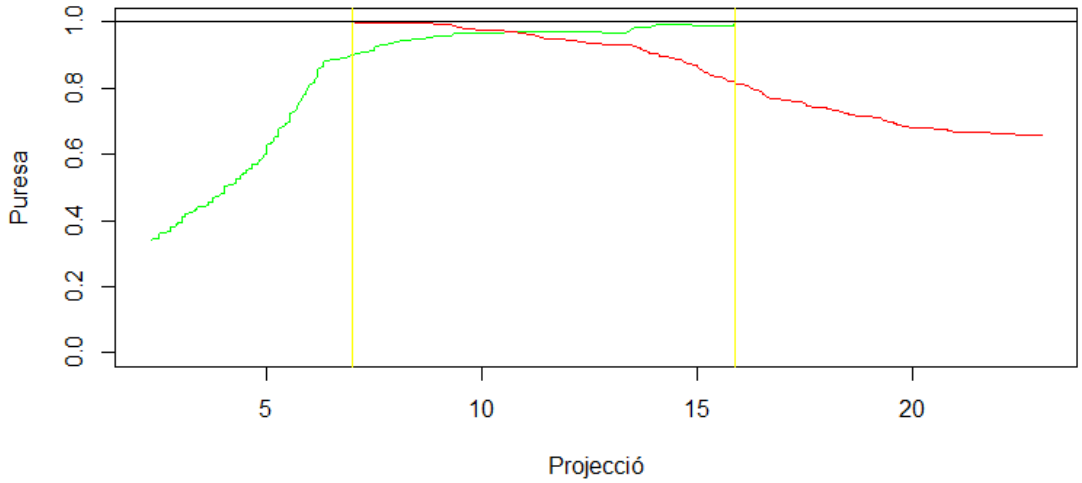

Figura 5.42: Puresa de les dades (dataset BC)

A la figura 5.42 podem apreciar que la puresa de les dades és bastant regular al llarg de la projecció realitzada. A més, la tria de puresa del 100% ja permet assolir una reducció del molt gran, retornant només al voltant del 20% de les dades originals.

Ara, entrenem una SVM amb el conjunt de dades filtrades. Els resultats es mostren a la taula 5.11, al final de l'apartat.

Ara, realitzarem una aproximació no lineal al dataset. Per a poder realitzar una comparació adequada, utilitzarem la mateixa configuració que Opposite Maps per al kernel RBF ($\sigma = 1.5$).

Si executem el nostre mètode amb el kernel especificat, el conjunt d'entrenament resulta separable linealment en Feature Space. Per aquesta raó, fixarem el nombre de dades de filtratge en un 50% de les dades originals.

Kernel	Mètode	Nº dades (%)	Error (%)
Lineal	–	100.00	3.0
Lineal	Opposite Maps	79.10	3.3
Lineal	Filtratge per puresa (projecció FDA)	18.49	3.0
RBF($\sigma = 1.5$)	–	100.00	2.9
RBF($\sigma = 1.5$)	Opposite Maps	59.80	4.0
RBF($\sigma = 1.5$)	Filtratge per puresa (projecció KFDA)	49.95	3.6

Taula 5.11: Resultats del dataset BC

Igual que amb el dataset anterior, els resultats milloren Opposite Maps tant a nivell de reducció com d'error. Concretament, amb una puresa del 100% s'assoleix el mateix nivell d'error que amb la totalitat de les dades, utilitzant-ne un nombre molt menor.

Aquest dataset, tot i que és més gran que l'anterior (683 mostres) tampoc té un nombre massa elevat de dades i encara és possible que els resultats es vegin condicionats per la tria dels subconjunts d'entrenament, validació i test.

5.3.8 Dataset PID

Per a realitzar les probes amb el dataset PID, s'ha realitzat una partició de les dades de tal manera que el 60% pertanyen ara al conjunt d'entrenament, el 20% al conjunt de test i el 20% restant al de validació. A més, per tal de poder realitzar una comparació amb els resultats del mètode *opposite maps*, s'utilitzaran els mateixos paràmetres que en els seus experiments en la configuració dels kernel. El paràmetre de cost, però, s'ha hagut de validar independentment.

Com que estem treballant amb un conjunt de dades prou reduït, no realitzarem fase de projecció en cap del casos. Per començar, aplicarem la versió lineal del mètode, amb una pures del 100% per d'obtenir una primera impressió del dataset.

El resultat de la projecció es pot trobar a la figura 5.43. En aquest cas, la projecció de les dades és molt més irregular que en els altres dos casos: concretament, la classe positiva segueix una distribució relativament gaussiana, però la classe negativa té una distribució clarament formada per una barreja de gaussianes.

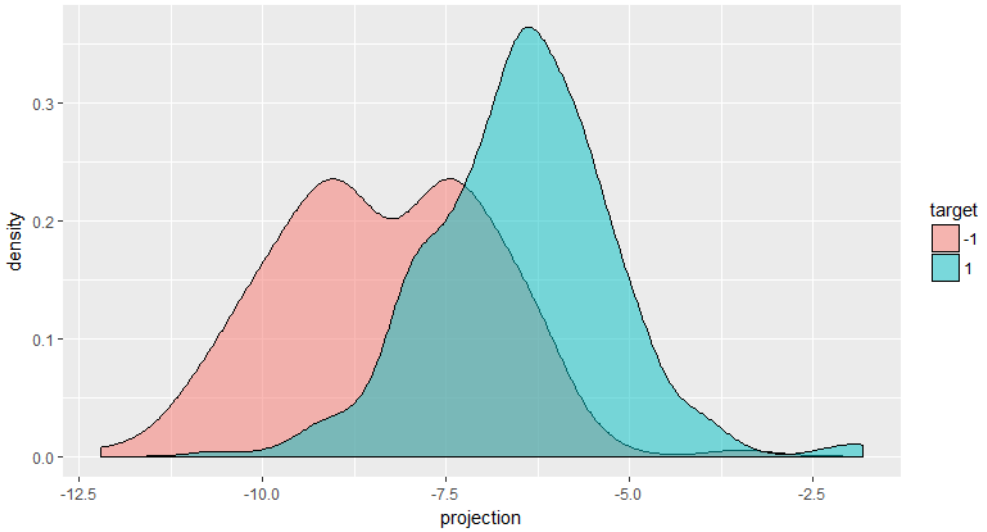

Figura 5.43: Projecció lineal de les dades (dataset PID)

A continuació, seleccionem les dades utilitzant un 100% de puresa.

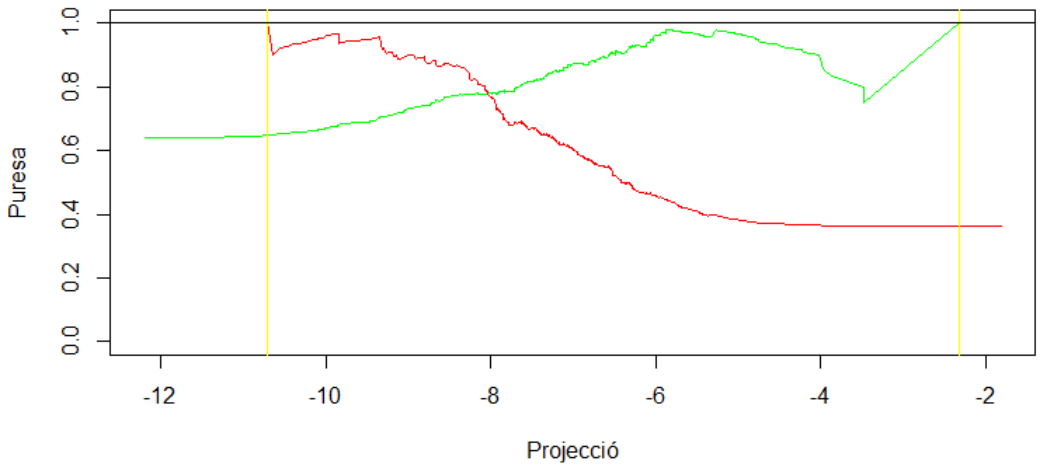

Figura 5.44: Puresa de les dades (dataset PID – 100% puresa)

En aquest cas, però, el solapament de les dades és major, i només obtenim una reducció del 2.16%. Això ens indica que hem de disminuir el valor del coeficient de puresa per tal d'obtenir una selecció més àmplia.

En aquest cas, amb un valor de puresa del 90% obtenim un conjunt de dades del 43.29% del total. Així, amb una lleugera disminució del coeficient de puresa, hem eliminat moltes de les dades situades que podrien ser considerades *outliers*, ja que es troben en zones amb molta proporció de dades de la classe contraria.

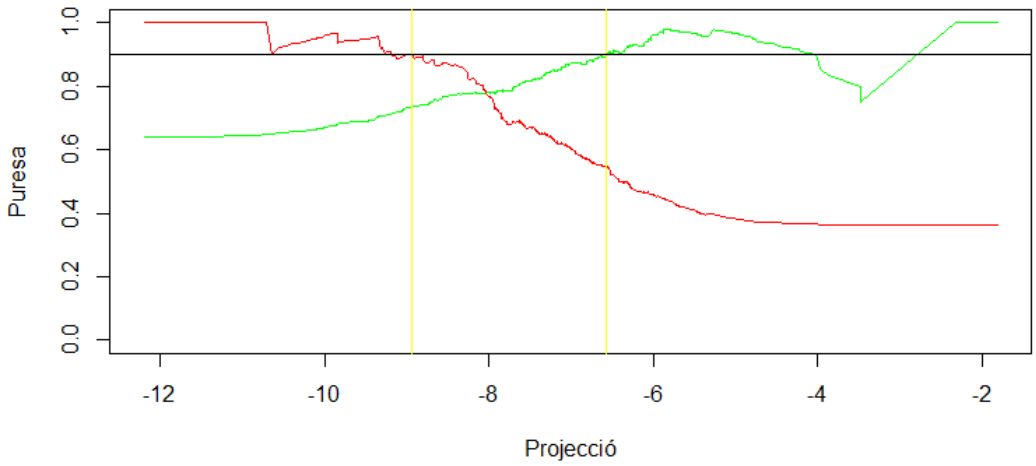

Figura 5.45: Puresa de les dades (dataset PID – 90% puresa)

Ara, entrenem una SVM amb el conjunt de dades filtrades. Els resultats es mostren a la taula 5.12, al final de l'apartat.

A continuació, realitzarem una aproximació no lineal al dataset. Per a poder realitzar una comparació adequada, utilitzarem la mateixa configuració que Opposite Maps per al kernel RBF ($\sigma = 1.5$).

Si executem el nostre mètode amb el kernel especificat, el conjunt d'entrenament resulta separable linealment en Feature Space. Si fixem el nombre de dades de filtratge en un 50% de les dades originals, els resultats obtinguts amb la svm filtrada no són prou bons (amb errors al voltant del 40%). Per aquesta raó, en aquest exemple s'ha incrementat el nombre de dades de filtratge fins al 80%.

Kernel	Mètode	Nº dades (%)	Error (%)
Lineal	–	100.00	23.1
Lineal	Opposite Maps	91.76	23.4
Lineal	Filtratge per puresa (projecció FDA)	43.29	23.1
RBF ($\sigma = 1.5$)	–	100.00	22.8
RBF ($\sigma = 1.5$)	Opposite Maps	89.4	24.5
RBF ($\sigma = 1.5$)	Filtratge per puresa (projecció KFDA)	80.00	24.2

Taula 5.12: Resultats del dataset PID

Igual que en els altres datasets, el nostre mètode millora Opposite Maps tant a nivell de reducció com d'error. En aquest, però, la complexitat del dataset dificulta la utilització del kernel RBF, que requereix un gran nombre de dades per a obtenir resultats raonables.

5.3.9 Dataset CCFD

Degut al gran nombre de dades del CCFD, la partició del dataset s'ha realitzat de tal manera que el 50% de les dades pertany al conjunt d'entrenament, el 25% al de test i el 25% restant al de validació. En aquest cas, s'han de validar tots els paràmetres per tal d'obtenir el millor resultat possible.

Per tal de poder comparar millor els resultats en un dataset tan poc balancejat, a més dels resultats de cada execució es mostrarà l'anomenada *confusion matrix*, una taula que ens indica el nombre de prediccions correctes i incorrectes, segons la classe a la que haurien d'haver estat classificades.

La primera aproximació ha estat executar una SVM amb un kernel lineal. Per a aquesta execució, s'ha validat el paràmetre de cost amb valors $C = \{2^{-3}, \dots, 2^{3}\}$. Els resultats es mostren a la taula 5.13 i a la taula 5.15.

A continuació, s'ha iniciat el mètode de projecció per puresa. L'execució del mètode amb un coeficient de puresa del 100% retorna un 59.25% de les dades, fet que indica que probablement es podrà realitzar una bona classificació. Tot i això, el nombre de dades és molt elevat encara, de manera que s'ha fixat la mida de les dades de sortida al 40% de les dades originals.

Degut a la gran diferència entre el nombre d'elements de cada classe (els casos de frau són només el 0.172%), en els dos casos anteriors s'ha afegit el conjunt de dades addicionals, que inclou tots els elements de la classe *Fraud* que no apareixen al conjunt retornat. D'aquesta manera, el filtratge de dades es produeix només a la classe majoritària, i no a la minoritària.

El paràmetre de cost per a la SVM entrenada amb aquestes dades s'ha validat igual que el de la SVM anterior, amb un rang de valors $C = \{2^{-3}, \dots, 2^{3}\}$. Els resultats d'ambdós casos es poden veure també a la taula 5.13 i a la 5.15.

Dades utilitzades		100 %		59.25 %		40.00 %	
Valor predit		*Fraud*	*Legit*	*Fraud*	*Legit*	*Fraud*	*Legit*
Valor real	*Fraud*	97	34	103	28	96	35
	Legit	14	71056	19	71051	15	71055

Taula 5.13: Confusion Matrix *del dataset CCFD amb kernel lineal.*

Els resultats obtinguts amb aquests conjunts filtrats són molt semblants als obtinguts utilitzant la totalitat de les dades, i fins i tot en algun cas els resultats són lleugerament superiors a nivell d'error. En qualsevol cas, la reducció del temps d'entrenament ha estat significativa (fins a 12 vegades més ràpid utilitzant només un 40% de les dades).

Ara, realitzarem diverses probes amb kernels no lineals. Per començar, utilitzarem el kernel quadràtic. Per a aquesta execució, s'ha validat el paràmetre de cost amb valors $C = \{2^{-3}, \dots, 2^{3}\}$. Els resultats es mostren a la taula 5.14 i a la taula 5.15.

Cal remarcar que l'execució del mètode amb el kernel quadràtic ha estat significativament més ràpida. Això pot ser degut a l'estructura de les dades, que és més senzilla en el Feature Space induït per aquest kernel. També el seu error de generalització és major, segurament degut a un sobreajust a les dades d'entrenament.

A continuació, s'ha iniciat el mètode de projecció per puresa. La primera aproximació s'ha realitzat mitjançant la projecció per KFDA, utilitzant 1000 representants de cada classe i k-means per a la reducció. A més, s'ha computat la mitjana de 100 projeccions diferents.

Tot i això, els resultats del mètode no han estat els esperats: el nivell d'error és al voltant del 1 %, molt elevat per al nostre dataset. Per aquesta raó, s'ha decidit aplicar la projecció per SVM. En aquest cas, s'ha validat un paràmetre de cost de la SVM de la projecció utilitzant el conjunt de validació. La resta de paràmetres de configuració han estat idèntics als de la projecció via KFDA.

Els l'execució del mètode amb la projecció via SVM mostra resultats molt similars als de KFDA, que no són satisfactoris. És possible que això sigui degut no a la projecció, sinó al conjunt de representants, que potser és massa petit (menys de l'1% del total). El problema és que els dos mètodes anteriors tenen un cost massa elevat per a poder-se executar amb un major nombre de representants. Per aquesta raó, es valora la utilització de la projecció per centroides, que és molt més ràpida i podrà utilitzar un major nombre de representants (10000 en aquest cas).

El paràmetre de cost per a la SVM entrenada amb aquestes dades s'ha validat igual que el de la SVM anterior, amb un rang de valors $C = \{2^{-3}, \dots, 2^{3}\}$. Els resultats es poden veure a les taules 20 i 21.

Dades utilitzades		100 %		60% – Centroides		75% – Centroides	
Valor predit		*Fraud*	*Legit*	*Fraud*	*Legit*	*Fraud*	*Legit*
Valor real	*Fraud*	94	37	95	36	96	35
	Legit	29	71041	65	71005	47	71023

Taula 5.14: Confusion Matrix *del dataset CCFD amb kernel quadràtic.*

Kernel	Configuració	Nº dades (%)	Cost	Error (%)	Temps
Lineal	–	100.00	2^{-3}	0.06741	6 h
Lineal	Projecció FDA	59.19	2^{-3}	0.06601	1 h
Lineal	Projecció FDA	40.00	2^{-3}	0.07022	32 min
Quadràtic	–	100.00	2^{-3}	0.09269	28 min
Quadràtic	Projecció Centroides	60.00	2^{-3}	0.14185	8 min
Quadràtic	Projecció Centroides	75.00	2^{-3}	0.11517	12 min

Taula 5.15: Resultats del dataset CCFD

5.3.9 Dataset LR

Per al dataset LR, la partició del dataset s'ha realitzat de tal manera que el 60% de les dades pertany al conjunt d'entrenament, el 20% al de test i el 20% restant al de validació. En aquest cas, igual que amb el dataset anterior, s'han de validar tots els paràmetres per tal d'obtenir el millor resultat possible.

Igual que a l'apartat anterior, a més dels resultats de cada execució es mostrarà la *confusion matrix* de cada SVM entrenada a les dades de test.

La primera aproximació ha estat executar una SVM amb un kernel lineal. Com que aquest dataset té una mida més reduïda, per a aquesta execució s'ha validat el paràmetre de cost amb valors $C = \{2^{-3}, \dots, 2^{3}\}$. Els resultats es mostren a la taula 5.16 i a la taula 5.18.

Cal remarcar que el classificador lineal no ha estat capaç de separar els conjunts de dades degut a la seva elevada complexitat: el classificador assigna a totes les dades la classe *Consonant*, independentment del seu valor. Tenint en compte aquest fet, es valorarà si l'aplicació del nostre mètode pot compensar aquest fet d'alguna manera.

Així, l'execució del mètode amb un coeficient de puresa del 100% retorna un 84.17% de les dades. Com que aquest nombre és molt elevat, es realitzarà també una execució utilitzant només el 50% de les dades originals.

El paràmetre de cost per a la SVM entrenada amb aquestes dades s'ha validat igual que el de la SVM anterior, amb un rang de valors $C = \{2^{-3}, \dots, 2^{3}\}$. Els resultats d'ambdós casos es poden veure també a la taula 5.16 i a la 5.19.

Dades utilitzades		100 %		84.38 %		50 %	
Valor predit		*Cons.*	*Vowel*	*Cons.*	*Vowel*	*Cons.*	*Vowel*
Valor real	*Cons.*	3198	0	3198	0	3198	0
	Vowel	802	0	802	0	802	0

Taula 5.16: Confusion Matrix *del dataset LR amb kernel lineal.*

En vista dels resultats obtinguts, podem concloure que el conjunt no és separable linealment, i que el nostre algoritme no pot compensar aquest fet.

Ara, utilitzarem una SVM amb kernel RBF. Igual que abans, hem validat el paràmetre de cost amb valors $C = \{2^{-3}, \dots, 2^{3}\}$. Els resultats es mostren a la taula 5.17 i a la taula 5.19.

En aquest cas, el rendiment és molt major, ja que el kernel RBF és capaç de separar adequadament les dades. Per a aquest filtratge, s'ha utilitzat l'algoritme de k-means com a algoritme de reducció amb 1200 dades de cada classe, un 20% del total. Aquesta xifra és una mica elevada en comparació amb altres datasets, però és necessari per a assolir un bon rendiment.

Per a l'entrenament de les SVM amb les dades filtrades, s'han plantejat quatre casos diferents: Amb la puresa del 100% i amb només un 50 % de les dades, per a projecció via KFDA i SVM. Per al paràmetre de cost de la SVM, s'ha validat igual que s'ha fet als apartats anteriors. Els resultats es mostren a les taules 5.17, 5.18 i 5.19.

Dades utilitzades		100 %		84.38 %		50.00 %	
Valor predit		*Cons.*	*Vowel*	*Cons.*	*Vowel*	*Cons.*	*Vowel*
Valor real	*Cons.*	3194	4	3191	7	3187	11
	Vowel	147	655	201	601	205	597

Taula 5.17: Confusion Matrix *del dataset LR amb kernel RBF (KFDA).*

Dades utilitzades		100 %		84.38 %		50.00 %	
Valor predit		*Cons.*	*Vowel*	*Cons.*	*Vowel*	*Cons.*	*Vowel*
Valor real	*Cons.*	3194	4	3193	5	3192	10
	Vowel	147	655	195	607	200	598

Taula 5.18: Confusion Matrix *del dataset LR amb kernel RBF (SVM).*

Kernel	Configuració	Nº dades (%)	Cost	Error (%)	Temps
Lineal	–	100.00	2^{-3}	20.05	5 min
Lineal	Projecció FDA	84.38	2^{-3}	20.05	50 seg
Lineal	Projecció FDA	50.00	2^{-3}	20.05	20 seg
RBF ($\gamma = 1$)	–	100.00	2^{1}	3.78	10 min
RBF ($\gamma = 1$)	Projecció KFDA	64.67	2^{1}	5.20	3 min
RBF ($\gamma = 1$)	Projecció KFDA	50.00	2^{1}	5.40	2 min
RBF ($\gamma = 1$)	Projecció SVM	64.67	2^{1}	5.00	3 min
RBF ($\gamma = 1$)	Projecció SVM	50.00	2^{1}	5.25	2 min

Taula 5.19: Resultats del dataset LR

En aquest cas, podem veure com una reducció de casi el 50% de les dades implica només una pèrdua del rendiment de poc més de l'1%. A més, el temps s'ha reduït també a menys de la meitat, fet que pot ser molt avantatjós per a datasets més grans que aquest.

6. Conclusions

6.1. Assoliment de la planificació

Segons la planificació realitzada a l'apartat 2.1, aquest es divideix en tres grans fases:

- **Planificació del projecte**: Aquesta és la primera etapa a realitzar, i es desenvolupa sota la tutela dels professors de l'assignatura de GEP. Un cop finalitzada aquesta part, es podrà procedir a la realització del projecte en sí. Aquesta fase té una duració aproximada d'un mes, des del 18/09 fins al 23/10.

- **Execució del projecte**: En aquesta fase es realitza el gruix del treball: S'ha de realitzar la totalitat del disseny de l'algoritme o algoritmes a presentar, amb les seves corresponents implementacions i els resultats dels mateixos enfront de diversos jocs de probes. Aquesta fase té una duració aproximada d'un mes i mig, des del 24/10 fins al 18/12.

- **Redacció de la documentació**: En aquesta fase, cal documentar detalladament els algoritmes resultants de l'execució del projecte. A més cal recollir també en un document els resultats de les probes realitzades i un pseudocodi que representi adequadament els algoritmes dissenyats. Aquesta fase té una duració aproximada de dues setmanes, des del 19/12 fins al 31/12.

En el curs del projecte, aquest ha estat el grau d'assoliment de cadascuna de les fases:

- **Planificació del projecte**: Aquesta fase ha finalitzat sense problema dins del termini establert i amb l'assoliment de tots els objectius proposats.

- **Execució del projecte**: Aquesta fase ha finalitzat dues setmanes més tard del que estava previst, ja que s'han desenvolupat una gran quantitat d'idees per a millorar cada vegada més la qualitat i el cost temporal de les solucions obtingudes, i no s'ha volgut finalitzar la fase fins a haver pogut desenvolupar-les i validar-les totes. Per aquesta raó, la data final de tancament d'aquesta fase ha estat el 31/12.

- **Redacció de la documentació**: Aquesta fase ha finalitzat sense problema dins el termini establert, que era de dues setmanes. En allargar la fase anterior fins el 31/12, S'ha desplaçat el període d'aquesta, essent iniciada el 01/01 i finalitzada el 15/01. Tota la documentació s'ha realitzat sense cap problema en base als resultats obtinguts durant l'execució del projecte.

En general, cal remarcar que no s'han pogut valorar alguns dels objectius addicionals, ja que s'ha preferit explorar a fons totes les opcions disponibles per a resoldre els objectius principals. A més, ha faltat temps per a realitzar l'experimentació amb un nombre més gran de datasets, sobretot amb aquells de mida tal que ja no sigui viable entrenar una SVM estàndard utilitzant la totalitat de les dades disponibles.

6.2. Seguiment de la metodologia

La metodologia definida a l'apartat 2.2. determina que la feina es realitza en iteracions de dos cicles, que funcionen de la manera següent:

1. Es realitza una cerca bibliogràfica de diversos mètodes actuals per a la resolució del problema.

2. Es dissenya un esbós d'un algoritme que realitzi la tasca desitjada.

3. Es divideix l'algoritme anterior en el major nombre de fases diferenciades possible.

4. S'analitza cada fase per separat, tot cercant bibliografia que ajudi a resoldre el problema concret que es troba. S'intenta realitzar una optimització de la fase, tant a nivell de temps d'execució com de qualitat de la solució trobada.

5. Es construeix un prototipus de l'algoritme i es valora el seu rendiment, tant a nivell de temps d'execució com de qualitat de la solució trobada.

6. Si els resultats no són satisfactoris, tornar al pas 4.

7. Un cop el mètode ha estat optimitzat el màxim possible, es valoren alternatives per a cadascuna de les fases i es torna al pas 4.

8. Un cop s'han explorat totes les alternatives viables per a les diverses fases de l'algoritme, es torna al pas 1 i es pensa algoritmes alternatius per a realitzar la feina.

9. Un cop s'han ideat suficients algoritmes i s'han optimitzat per fases, es realitza una comparació global d'entre totes les opcions i es valoren les més viables, eficients i amb menor cost temporal.

Quan s'ha arribat a la data teòrica de finalització de la fase d'execució del projecte, s'ha valorat la feina que s'estava realitzant, i s'ha vist que amb un temps addicional es podrien obtenir molt millors resultats. Com que la planificació del projecte ja es va realitzar de tal manera que es disposés d'un període addicional de dues setmanes per si sorgien imprevistos, s'ha decidit allargar aquesta fase precisament dues setmanes més. D'aquesta manera, s'ha pogut finalitzar la fase d'execució amb millors resultats dels que es disposava en el moment de finalització proposat a la planificació.

Independentment de les modificacions del calendari, s'han realitzat adequadament els cicles d'iteracions amb un rendiment excel·lent: S'han pogut valorar una gran quantitat de possibles mètodes, cadascun d'aquests s'ha optimitzat de manera molt eficient i s'han detectat tots els errors i inconvenients de manera molt prematura.

Aquesta metodologia, per tant, s'ha seguit de manera molt rigorosa, i no ha estat necessari modificar-la. Els bons resultats (tant en la variabilitat dels mètodes com en la exhaustivitat de la optimització de les seves parts) del projecte en són un clar indicador.

6.3. Anàlisi d'alternatives

En aquest treball s'han considerat diverses opcions per a una mateixa estructura d'algoritme. Així, del nostre algoritme genèric, només se'n diferencien les implementacions concretes en la tria realitzada en algunes de les fases de l'algoritme.

Cadascuna d'aquestes tries tenen costos espacials i temporals diferents, amb diversos nivells de rendiment. Per tant, a priori, no era possible determinar quina configuració seria la que millor funcionés. Per a poder prendre aquesta decisió, ha estat determinant l'etapa d'experimentació, que ha permès descartar aquelles opcions que no eren viables o no han donat bons resultats, i també analitzar els costos i resultats de cadascuna de les opcions.

Cal destacar que tots els algoritmes proposats tenen una forta base teòrica que en justifica l'aplicabilitat, fins i tot abans de realitzar l'experimentació. Així, tots els candidats podien, a priori, funcionar igual de bé, i oferir resultats igualment satisfactoris.

6.4. Integració de coneixements

A priori, els coneixements requerits per a aquest projecte eren aquells basats en el temari i coneixements associats a l'assignatura d'Aprenentatge Automàtic (APA), però ha acabat requerint coneixements de moltes altres branques: Per començar, han estat necessaris el disseny, anàlisi i optimització d'algoritmes d'assignatures com EDA i Algorísmia, però també han estat molt importants els coneixements obtinguts d'assignatures de la branca de matemàtiques, com M1 (Àlgebra), M2 (Càlcul), CN (Computació Numèrica) o PE (Probabilitat i Estadística). A més, bona part del coneixement d'APA s'ha complementat amb aquell obtingut de l'assignatura de Mineria de Dades (MD), que aporta una altra visió de les tècniques estudiades.

La part de les assignatures d'algorísmia és necessària per al disseny adient de l'algoritme. El fet que haguem imposat una restricció de cost a l'algoritme requereix un coneixement en profunditat del càlcul de costos i del disseny d'algoritmes adients. A més, molts coneixements de Càlcul (sobre tot optimització) s'han aplicat al disseny de l'algoritme.

La vessant més matemàtica és necessària per a la implementació eficient dels càlculs proposats. En concret, en algunes fases s'ha de realitzar el càlcul de matrius pseudo-inverses de matrius simètriques i semi-definides positives. Aquest càlcul és molt lent si es realitza de la manera normal (amb la llibreria estàndard), però utilitzant els coneixements d'Àlgebra i de Computació Numèrica s'ha trobat una variant del mètode de Txoleski que permet realitzar aquest càlcul de manera molt més eficient.

Per últim, cal destacar que molts dels coneixements de MD i de APA s'han aplicat també a la fase de reducció de les dades, on diversos dels algoritmes de clustering estudiats s'han utilitzat amb èxit amb aquesta finalitat.

6.5. Assoliment d'objectius

A l'apartat 1.4. s'ha especificat una llista d'objectius a assolir en aquest projecte. En aquest apartat es repetiran aquests objectius, i se'n valorarà l'assoliment:

1. Dissenyar un (o diversos) **algoritmes eficients** que permetin realitzar una tria de les dades d'entrada, de manera que es pugui entrenar una SVM (per a classificació binària) amb aquestes dades i obtenir resultats el més semblants possibles als d'entrenar la mateixa SVM amb les dades originals.

   Aquest objectiu ha estat assolit en la seva totalitat, ja que s'ha dissenyat un mètode configurable que permet que una SVM sigui entrenada amb les dades resultants del mètode, i els resultats experimentals confirmen que, amb la configuració adequada, s'obtenen resultats molt semblants als de la SVM entrenada amb les dades originals.

2. Generar un **model predictiu** de manera eficient obtingut a partir del mateixos mètodes utilitzats per al filtratge de les dades.

   Aquest objectiu també s'ha assolit adequadament, ja que el mètode proposat retorna alhora un filtratge de les dades i un model predictiu basat en l'estimació del vector *w* de l'hiperplà de la SVM.

3. Obtenir un **model probabilístic** a partir dels models predictius generats.

   Aquest objectiu s'ha assolit conjuntament amb l'anterior, ja que amb una senzilla adaptació del model predictiu generat es pot aconseguir una sortida probabilística.

4. Validar **experimentalment** els diversos algoritmes dissenyats, i triar-ne aquell que mostri un millor rendiment, tant a nivell de cost espacial/temporal com a nivell de resultats.

   Aquest objectiu s'ha acomplert mitjançant el disseny experimental de l'apartat 5. En aquest apartat s'han validat per separat les diferents configuracions del mètode, tot triant-ne les configuracions que donen millors resultats, o que tenen un cost temporal més reduït.

5. Comparar l'algoritme triat amb **altres mètodes** que realitzin una tasca semblant, i **millorar-ne** els resultats.

   Aquest objectiu també s'ha assolit mitjançant el treball fet a l'apartat 5, on s'ha comparat el mètode proposat amb la tècnica de Opposite Maps en els mateixos datasets que es van utilitzar en l'article que el descriu. En general, el mètode proposat millora el rendiment de Opposite Maps en tots els datasets utilitzats, tant a nivell d'error generalitzat com a nivell de reducció de les dades.

6. Estendre l'algoritme dissenyat per a resoldre problemes de **classificació múltiple**.

   Aquest objectiu no s'ha pogut realitzar explícitament, en el sentit que no s'ha disposat de temps suficient per a desenvolupar una versió específica del mètode proposat per al problema de la classificació múltiple.

   D'altra banda, una de les maneres d'entrenar una SVM per al problema de la classificació múltiple és entrenar un seguit de classificadors binaris, que combinats entre sí permeten realitzar la classificació múltiple. En aquest sentit, es pot aplicar el mètode proposat a cadascun d'aquests classificadors entrenats, però tampoc s'ha disposat de prou temps per a validar aquesta opció.

7. Estendre l'algoritme dissenyat per a resoldre problemes de **regressió**.

   Aquest objectiu tampoc s'ha pogut realitzar, degut a la profunditat en que s'han explorat totes les possibles opcions. Tot i això, en vista del bon rendiment del mètode per a la classificació binària, sembla viable poder adaptar l'algoritme al problema de la regressió.

En general, podem concloure que el treball ha assolit sobradament els seus objectius principals, i fins i tot ha aconseguit assolir alguns dels objectius secundaris. Els objectius que no s'han assolit no s'han pogut realitzar per raons de calendari, de manera que és possible realitzar-los fora de l'àmbit d'aquest treball, com a complement del mateix.

6.6. Conclusions

El treball exposat en aquest document aporta una solució adequada i completa al problema plantejat: Com s'ha justificat a la part experimental, l'algoritme proposat permet entrenar una SVM només amb un subconjunt de les dades, reduint de manera significativa el cost de l'entrenament. A més, en molts casos els resultats són pràcticament igual de bons que els d'una SVM entrenada amb la totalitat de les dades disponibles.

Respecte de la planificació realitzada, el projecte s'ha pogut realitzar sense problemes dins del termini previst per a la seva execució. La única modificació realitzada a la planificació temporal ha estat la prolongació de la fase d'execució en dues setmanes addicionals, per tal de valorar algunes alternatives prometedores per a les diferents fases del nostre algoritme.

Aquesta prolongació de la fase d'execució no ha tingut cap efecte negatiu per al desenvolupament del projecte, ja que a la planificació inicial ja es va preveure que aquesta situació es podria donar, i es van reservar dues setmanes de temps addicional al final del projecte per si era necessari destinar-les a alguna de les etapes anteriors.

Aquesta modificació, per tant, no ha suposat cap despesa econòmica addicional, ja que estava prevista a la planificació realitzada a l'inici del projecte.

En aquest treball, s'ha realitzat una revisió en profunditat del funcionament dels classificadors binaris que utilitzen hiperplans, de les *Support Vector Machines* (SVM) i dels mètodes kernel en general. S'han comentat també altres tècniques menys conegudes, com l'algoritme de *Kernel Fisher Discriminant Analysis* (KFDA) que s'ha utilitzat com una part integrada del mètode de filtratge per puresa.

Un cop revisats tots els coneixements previs necessaris per al desenvolupament de l'algoritme proposat, se'n realitza una descripció detallada: Es defineix el concepte de puresa, es relaciona amb els hiperplans separadors òptims de les SVM i es presenta un algoritme capaç de realitzar la tasca requerida alhora que satisfà totes les condicions imposades a l'apartat d'objectius del projecte.

En l'explicació de l'algoritme de filtratge per puresa se n'exposen les diferents fases de manera completament independent. Aquesta estructura de l'algoritme el fa molt modulable, ja que no només es disposa d'un gran ventall de configuracions sinó que fins i tot és ampliable a altres opcions que no es contemplen en aquest document.

Un cop revisades les diferents etapes de l'algoritme, se'n realitza una visió general i es realitza un estudi en profunditat dels seus costos espacials i temporals. A més, l'anàlisi teòric dels costos va acompanyat també d'un anàlisi teòric del funcionament de les diferents fases. En aquest anàlisi teòric es justifica que les decisions preses estan orientades sempre a millorar el rendiment de l'algoritme, en base a la qualitat de les aproximacions realitzades. A més, es demostra que en certes condicions, l'algoritme sempre té el comportament esperat i es relaciona el valor del coeficient de puresa proporcionat per l'usuari amb la distribució de les dades a Feature Space i amb el conjunt de dades filtrades pel mètode.

Cal destacar que el problema proposat és molt més senzill de resoldre en Input Space que en Feature Space. Per aquesta raó, s'ha desenvolupat també una versió lineal de l'algoritme, que té un temps d'execució molt menor que la versió genèrica i ofereix els mateixos resultats si s'utilitza en combinació amb una SVM amb kernel lineal.

A la part experimental, s'han validat satisfactòriament les diferents opcions per a cadascuna de les fases, i s'ha arribat a les conclusions següents:

1) Dels algoritmes de reducció proposats, k-means i k-medoids són els que funcionen millor experimentalment. El primer és més ràpid, i el segon tendeix a ser més estable i oferir resultats lleugerament superiors. Per tant, la idea és la següent, en funció de la mida del dataset a utilitzar:

    a. Si el dataset és petit, utilitzar la totalitat de les dades (és a dir, no realitzar cap mena de reducció).

    b. Si el dataset és gran, utilitzar k-medoids.

    c. Si el dataset és molt gran, utilitzar k-means.

El nombre de representants a utilitzar també s'haurà de triar en funció de la mida del dataset, i del temps de computació del que es disposi.

2) Independentment de l'algoritme de reducció utilitzat, es proposen tres alternatives per a la configuració de la resta de l'algoritme:

    a. La primera opció a utilitzar és la combinació de **projecció per KFDA** i la **classificació per LDA**. Aquest mètode ha demostrat tenir un rendiment acceptable, i no requereix de la validació de cap paràmetre addicional.

    b. Si la opció anterior no aporta resultats prou satisfactoris, es pot utilitzar la combinació de **projecció per SVM** i la **classificació per SVM1D**. Aquesta combinació sol resultar en una millora de la qualitat del conjunt de sortida respecte a la proposta anterior. L'únic inconvenient és que és necessari validar un paràmetre de cost per a la SVM interna que utilitza l'algoritme. Aquest paràmetre, però, es pot validar de manera molt ràpida i senzilla utilitzant un conjunt de validació i els passos següents:
        i. Executem el mètode amb qualsevol valor de puresa, i conservem únicament el classificador per a cada valor del paràmetre de cost.
        ii. Validem el rendiment de cadascun dels classificadors obtinguts.
        iii. Realitzem el filtratge amb el valor de puresa desitjat i utilitzant el valor del paràmetre de cost que hagi generat el classificador amb millor rendiment en el conjunt de validació.

        D'aquesta manera, es pot validar el paràmetre de cost sense haver d'entrenar les diverses SVM amb el conjunt de dades filtrades.

    c. Si el conjunt de dades és massa gran, es pot donar el cas que sigui necessari un conjunt molt gran de representants. Si no n'agafem prou, la solució aportada degenerarà ràpidament, donant lloc a resultats molt dolents, però si n'agafem massa les propostes anteriors tindran un temps d'execució massa elevat. Per aquesta raó, quan el conjunt de dades sigui massa gran, es proposa utilitzar la **projecció per centroides** i la **classificació per LDA**. Aquesta combinació és la que té el menor cost espacial i temporal, i a més té l'avantatge de no requerir cap paràmetre de configuració addicional (com el paràmetre de cost de la projecció per SVM).

3) El conjunt de dades addicional (ADD) utilitzat per a balancejar les classes generalment millora el rendiment del classificador entrenat. Per tant, la primera opció serà sempre entrenar només amb el conjunt de dades filtrades, i si el rendiment no és satisfactori aleshores afegir-hi el conjunt addicional. En cap cas s'ha d'utilitzar el conjunt de dades de la banda contraria del marge (OUT), ja que provoca una caiguda dràstica del rendiment de la SVM entrenada.

També s'ha validat també el funcionament de l'algoritme en comparació amb un altre mètode que realitza la mateixa tasca, *Opposite Maps*. En pràcticament tots els casos, el nostre mètode ofereix un rendiment superior al de *Opposite Maps*, tant a nivell de reducció com a nivell d'error obtingut a la classificació. A més, s'han realitzat probes en datasets de mida més gran, per a veure el rendiment del nostre mètode en casos on comença a no ser viable la utilització d'una SVM amb la totalitat de les dades.

En general, podem concloure que els objectius d'aquest treball s'han assolit satisfactòriament, i que els resultats experimentals obtinguts confirmen que el mètode de filtratge per puresa és una opció raonable a utilitzar en aquells casos en que el nombre de dades és massa gran per a poder realitzar l'entrenament d'una SVM.

6.7. Treball futur

La finalització d'aquest projecte, tot i que amb resultats satisfactoris, deixa encara un seguit de problemes sense resoldre. Alguns d'aquests problemes no s'han pogut resoldre durant l'execució del projecte i no s'han inclòs al mètode, i d'altres no s'han pogut plantejar en profunditat, però amb una major quantitat de temps es podrien arribar a desenvolupar sense problemes.

1) **Millorar el rendiment de KFDA**: La versió de KFDA proposada és una derivació directa de l'aplicació de FDA a Feature Space, i aplicant el *kernel Trick*. Aquesta versió, però, es pot optimitzar per a assolir solucions en amb un cost espacial i temporal molt menor utilitzant una estratègia *greedy* [5]. A més, aquesta versió de l'algoritme millora la *sparsity* de la solució, ja que la versió proposada de KFDA utilitza tots els vectors per al càlcul de les projeccions i la versió *greedy* només n'utilitza un subconjunt.

2) **Filtratge per puresa per a classificació múltiple**: Tal i com s'ha explicat a l'apartat de l'assoliment d'objectius, aquest mètode es pot aplicar per separat a cadascun dels classificadors binaris utilitzats per a la classificació múltiple. Queda pendent, però, el disseny d'una versió de l'algoritme de filtratge per puresa que retorni un únic subconjunt en el qual es puguin entrenar els diversos classificadors binaris utilitzats.

   Un possible punt de partida per a la resolució d'aquest problema podria ser el fet que l'algoritme de LDA es pot adaptar molt fàcilment a problemes de classificació múltiple. L'algoritme de LDA multi-classe retorna una direcció de projecció per a cada classe, que utilitza per a calcular per a cada nova dada la probabilitat de pertinència a cadascuna de les classes donades.

   Si utilitzem aquestes direccions, podem calcular per al vector w_k associat a la classe *k* la puresa de les dades de la classe *k* per sobre d'un hiperplà amb la direcció de w_k respecte de les dades de les altres classes. Així, podríem eliminar per a cada classe el subconjunt de dades més pur, i retornar només aquelles dades que es troben en una zona "central", és a dir, amb una menor puresa.

A la proposta anterior s'hauria d'aplicar el *Kernel Trick* per a obtenir una versió de l'algoritme que treballi a Feature Space, però l'estructura de l'algoritme de filtratge per puresa multi-classe serà molt semblant a la de l'algoritme de filtratge per puresa per a classificació binària.

3) **Filtratge per puresa per a regressió**: Adaptar l'algoritme de filtratge per puresa al problema de la regressió és més complicat, ja que la classificació binària i la regressió són problemes més diferenciats.

   Tot i això, una possible adaptació podria partir de la premissa següent: Si en el classificador utilitzem SVMs (binàries) o FDA per a estimar la direcció de separació de les dades, en aquest problema podem utilitzar SVMs (de regressió) o PCA per a estimar l'hiperplà que aproxima millor aquestes dades.

   En el cas de les SVMs l'adaptació és natural, com ha passat en el nostre algoritme. En el cas de l'aproximació per PCA, hem de trobar l'hiperplà que s'estén per totes les components excepte la de menor variància. D'aquesta manera, només resta construir els marges paral·lels a l'hiperplà obtingut per a realitzar un filtratge adequat.

   Un problema addicional per a adaptar l'algoritme al problema de la regressió és que les dades ja no tenen una classe assignada, de manera que no podem computar els coeficients de puresa tal i com ho hem realitzat en aquest treball. Per aquesta raó, és necessari re-definir el concepte de puresa per a dades que no tenen cap classe assignada. Aquesta definició podria estar basada en aspectes com la densitat de les dades o la seva distribució en la direcció de la projecció. També seria possible definir altres criteris diferents, més adequats al nou problema a tractar.

Referències

[1] Christopher M. Bishop. Pattern recognition and Machine Learning. Springer, 2006.

[2] Ethem Alpaydin. Introduction to Machine Learning. MIT press, 2014.

[3] Colin Campbell, Yiming Ying. Learning with Support Vector Machines. Morgan & Claypool, 2011.

[4] Corinna Cortes, Vladimir Vapnik. Support-Vector Networks. Kluwer, 1995.

[5] Bernhard Scholkopf, Alexander J. Smola. Learning with Kernels. MIT press, 2002.

[6] L. Bottou, O. Chapelle, D. DeCoste, J. Weston. Large Scale Kernel Machines. MIT press, 2007.

[7] Bernhard Scholkopf, Robert C. Williamson, Peter L. Bartlett. New Support Vector Algorithms. Neural Computation 12 1207–1245, 2000.

[8] Rimah Amami. Practical Selection of SVM Supervised Parameters with Different Feature Representations for Vowel Recognition. JDCTA, 2013.

[9] B. E. Boser, I. M. Guyon, V. N. Vapnik. A training algorithm for optimal margin classifiers. Proceedings of the fifth annual workshop on Computational learning theory - COLT '92, 1992.

[10] John C. Platt. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Microsoft Research, 1998.

[11] Ivor W. Tsamg, James T. Kwok, Pak-Ming Cheung. Very Large SVM Training using Core Vector Machines. Hong Kong University.

[12] Kuan-Ming Lin, Chih-Jen Lin. A Study on Reduced Support Vector Machines. Taiwan University.

[13] Yuh-Jye Lee, Olvi L. Mangasarian. RSVM: Reduced Support Vector Machines. University of Wisconsin.

[14] Son Lam Phung, Giang Hoang Nguyen, Abdsselam Bouzerdom. Efficient SVM training with Reduced Weighted samples. University of Wollongong, 2010.

[15] Benyang Tang, Dominic Mazzoni. Multiclass Reduced-Set Support Vector Machines. California Institute of Technology, 2006.

[16] DucDung Nguyen, TuBao Ho. An Efficient Method for Simplifying Support Vector Machines. Japan Advanced Institute of Science and Technology.

[17] Ajalmar R. Rocha Neto, Guilherme A. Barreto. Opposite Maps: Vector Quantization Algorithms for Building Reduced-Set SVM and LSSVM Classifiers. Federal University of Ceará.

[18] Carlos E. Pedreira. Learning Vector Quantization with Training Data Selection. Federal University of Rio de Janeiro, 2005.

[19] R. T. Peres, C. E. Pedreira.Generalized Risk Zone: Selecting Observations for Classification. Federal University of Rio de Janeiro, 2009.

[20] Robert Jenssen, Jose C. Principe, Deniz Erdogmus, Torbjorn Eltoft. The Cauchy-Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels. Elsevier, 2006.

[21] Marc G. Genton. Classes of Kernels for Machine Learning: A Statistics Perspective. Journal of Machine Learning Research 2, 2001.

[22] Alexander J. Smola, Peter Bartlett, Bernhard Schölkopf, Dale Schuurmans. Advances in Large Margin Classifiers. MIT press, 1999.

[23] Yang Su, T. M. Murali, Vladimir Pavlovic, Simon Kasif. Training Support Vector Machines in 1D. 1999.

[24] Johannes Schneider, Jasmina Bogojeska, Michail Vlachos. Solving Linear SVMs with Multiple 1D Projections. ACM, 2014.

[25] Richard O. Duda, Peter E. Hart, David G. Stork. Pattern Classification. Wiley-Interscience, 2001.

[26] Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopf, Klaus-Robert Müller. Fisher Discriminant Analysis with kernels. IEEE, 1999.

[27] Basat en la resposta de l'usuari math.stackexchange.com/users/90996 (*Algebraic Pavel*) a la pregunta *LDL decomposition and pseudoinverse*. Recuperat el 15 de Gener de 2018 de math.stackexchange.com/q/778414

[28] Donghui Yan, Ling Huang, Michael I. Jordan. Fast Approximate Spectral Clustering. ACM, 2009.

[29] A. Amnon Shashua. On the Equivalence Between the Support Vector Machine for Classification and Sparsified Fisher's Linear Discriminant. Hebrew University of Jerusalem, 1998.

[30] Jing Yang, Liya Fan. Weighted Generalized Kernel Discriminant Analysis Using Fuzzy Memberships. Transactions on mathematics, 2011.

[31] Ramu Yerukala, Naveen Kumar Boiroju. Approximations to Standard Normal Distribution Function. International Journal of Scientific & Engineering Research, 2015.

[32] UCI Machine Learning Repository. Recuperat el 15 de Gener de 2018 de archive.ics.uci.edu/ml

[33] Kaggle: The Home of Data Science & Machine Learning. Recuperat el 15 de Gener de 2018 de www.kaggle.com

[34] UCI Machine Learning Repository: Vertebral Column Pathologies. Recuperat el 15 de Gener de 2018 de archive.ics.uci.edu/ml/datasets/vertebral+column

[35] UCI Machine Learning Repository: Breast Cancer. Recuperat el 15 de Gener de 2018 de archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)

[36] UCI Machine Learning Repository: Pima Indian Diabetes. Recuperat el 15 de Gener de 2018 de archive.ics.uci.edu/ml/datasets/pima+indians+diabetes

[37] Kaggle: Credit Card Fraud Detection. Recuperat el 15 de Gener de 2018 de www.kaggle.com/dalpozz/creditcardfraud

[38] UCI Machine Learning Repository: Letter Recognition Data Set. Recuperat el 15 de Gener de 2018 de archive.ics.uci.edu/ml/datasets/Letter+Recognition